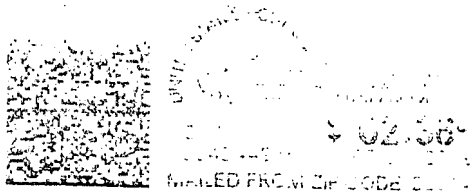
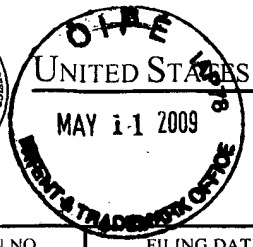
Organization TC2800 Bldg/Room Jeff
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
If Undeliverable Return in Ten Days

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300



AN EQUAL OPPORTUNITY EMPLOYER

DFW



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/667,010	09/21/2000	Uve Hansmann	IBM-116	8803

7590 04/30/2009

EXAMINER	
MOORTHY, ARAVIND K	
ART UNIT	PAPER NUMBER
2431	

DATE MAILED: 04/30/2009

Thomas A Beck
26 Rockledge Lane
New Milford, CT 06776

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 824 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 824 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

7590 04/30/2009

Thomas A Beck
26 Rockledge Lane
New Milford, CT 06776

EXAMINER	
MOORTHY, ARAVIND K	
ART UNIT	PAPER NUMBER
2431	

DATE MAILED: 04/30/2009

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/667,010	09/21/2000	Uve Hansmann	IBM-116	8803

TITLE OF INVENTION: HARDWARE-ORIENTED CONFIGURATION AND LOCKING OF DEVICES

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$0	$0	$1510	07/30/2009

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PTOL-85 (Rev. 08/07) Approved for use through 08/31/2010.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

7590 04/30/2009

Thomas A Beck
26 Rockledge Lane
New Milford, CT 06776

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/667,010	09/21/2000	Uve Hansmann	IBM-116	8803

TITLE OF INVENTION: HARDWARE-ORIENTED CONFIGURATION AND LOCKING OF DEVICES

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	NO	$1510	$0	$0	$1510	07/30/2009

EXAMINER	ART UNIT	CLASS-SUBCLASS
MOORTHY, ARAVIND K	2431	726-034000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 _____

2 _____

3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ☐ Individual ☐ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:

☐ Issue Fee

☐ Publication Fee (No small entity discount permitted)

☐ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

☐ A check is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

☐ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.

☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Notice of Allowability	Application No.	Applicant(s)	
	09/667,010	HANSMANN ET AL.	
	Examiner	Art Unit	
	ARAVIND K. MOORTHY	2431	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *16 December 2008*.

2. ☒ The allowed claim(s) is/are *1-15*.

3. ☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☒ All b) ☐ Some* c) ☐ None of the:
      1. ☒ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date *3/25/09* .
7. ☐ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

/Ayaz R. Sheikh/
Supervisory Patent Examiner, Art Unit 2431

DETAILED ACTION

1. This is in response to the communications filed on 16 December 2008.

2. Claims 1-15 are pending in the application.

3. Claims 1-15 have been allowed.

Allowable Subject Matter

4. Claims 1-15 are allowed.

The following is an examiner's statement of reasons for allowance:

The current application is directed towards setting basic means of access for operation of electronically operated devices, with the aid of a possibly transferable personal authentication system. The invention is essentially based on three components; namely additional device hardware functions permitting means of access, namely in particular for custom configuration and shutdown of the devices; a hardware-oriented interface to a reader device, for the authentication system such as a SmartCard reader permitting access to the said functions by a SmartCard; and the authentication system itself, capable of directly accessing the configuration and/or shutdown/startup/restart functions of the device hardware by way of the defined interface. Legitimization for configuration/shutdown and startup/restart of the devices is provided by matching of keys stored on the SmartCard and in a ROM in the device.

The closest prior art to the current application is Clark U.S. Patent No. 5,892,902. Clark is directed towards an intelligent token protected system includes a local host computer, an intelligent token in communication with the local host computer and a remote host computer in communication with the local host computer. The intelligent token interacts with the local host computer to perform a secure boot on the local host computer with minimal user input. Without

additional user input, the intelligent token also interacts with the remote host computer to authenticate the local host computer to the remote host computer.

However, there are differences between the current application and the Clark reference. Clark does not disclose the variety of devices that are claimed. There is no deactivation step disclosed by Clark. The current application deactivates the system to authorization patterns prior to operation. Clark is setting up a connection between computers, but is using a different security system to do so. In the current application, a key may be present on the device and the same key on the smart card, so a challenge/response can be used to authenticate the smart card. In the Clark reference, there is only a single level of access. The current application deals with different levels of access to differentiate the different levels of authentication or authorization that persons with different roles may need. Clark does not disclose "no access or full access and allow more finely defined levels of access".

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

5. inquiry concerning this communication or earlier communications from the examiner should be directed to ARAVIND K. MOORTHY whose telephone number is (571)272-3793. The examiner can normally be reached on Monday-Friday, 8:00-5:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Ayaz R. Sheikh can be reached on 571-272-3795. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Aravind K Moorthy/
Examiner, Art Unit 2431
/Ayaz R. Sheikh/
Supervisory Patent Examiner, Art Unit 2431

Examiner-Initiated Interview Summary	Application No.	Applicant(s)	
	09/667,010	HANSMANN ET AL.	
	Examiner	Art Unit	
	ARAVIND K. MOORTHY	2431	

All Participants:

(1) *ARAVIND K. MOORTHY*.

(2) *Thomas Beck*.

Status of Application: _____

(3) _____.

(4) _____.

Date of Interview: *25 March 2009*

Time: _____

Type of Interview:
☒ Telephonic
☐ Video Conference
☐ Personal (Copy given to: ☐ Applicant ☐ Applicant's representative)

Exhibit Shown or Demonstrated: ☐ Yes ☐ No
If Yes, provide a brief description:

Part I.

Rejection(s) discussed:
N/A

Claims discussed:
1

Prior art documents discussed:
N/A

Part II.

SUBSTANCE OF INTERVIEW DESCRIBING THE GENERAL NATURE OF WHAT WAS DISCUSSED:
The attorney discussed the case with the examiner. The examiner notified the attorney that the amendment put the case in condition for allowance.

Part III.

☒ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview directly resulted in the allowance of the application. The examiner will provide a written summary of the substance of the interview in the Notice of Allowability.

☐ It is not necessary for applicant to provide a separate record of the substance of the interview, since the interview did not result in resolution of all issues. A brief summary by the examiner appears in Part II above.

/Aravind K Moorthy/
Examiner, Art Unit 2431

(Applicant/Applicant's Representative Signature – if appropriate)

Notice of References Cited	Application/Control No. 09/667,010	Applicant(s)/Patent Under Reexamination HANSMANN ET AL.	
	Examiner ARAVIND K. MOORTHY	Art Unit 2431	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2008/0082983 A1	04-2008	Groetzner et al.	718/105
*	B	US-2008/0064367 A1	03-2008	Nath et al.	455/411
*	C	US-2008/0052505 A1	02-2008	Theobald, Holger	713/001
*	D	US-2008/0046529 A1	02-2008	Gilhuly et al.	709/206
*	E	US-2008/0024322 A1	01-2008	Riemschneider et al.	340/904
*	F	US-2006/0259769 A1	11-2006	Goettfert et al.	713/168
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Kulkarni et al, Context-Aware Role-based Access Control in Pervasive Computing Systems, 2008, ACM, pages 113-122
	V	Whittle et al, Executable Misuse Cases for Modeling Security Concerns, 2008, ACM, pages 121-130
	W	Maaser et al, Providing granted rights with anonymous certificates, 2008, IEEE, pages 890-893□□
	X	Michael Clifford, Networking in the Solar Trust Model: determining optimal trust paths in a decentralized trust network, 2002, IEEE, pages 271-281□□

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Context-Aware Role-based Access Control in Pervasive Computing Systems

Devdatta Kulkarni and Anand Tripathi*
Dept. of Computer Science, University of Minnesota
Twin Cities, MN 55455, USA
(dkulk,tripathi)@cs.umn.edu

ABSTRACT

In this paper we present a context-aware RBAC (CA-RBAC) model for pervasive computing applications. The design of this model has been guided by the context-based access control requirements of such applications. These requirements are related to users' memberships in roles, permission executions by role members, and context-based dynamic integration of services in the environment with an application. Context information is used in role admission policies, in policies related to permission executions by role members, and in policies related to accessing of dynamically interfaced services by role members. The dynamic nature of context information requires model-level support for revocations of role memberships and permission activations when certain context conditions fail to hold. Based on this model we present a programming framework for building context-aware applications, providing mechanisms for specifying and enforcing context-based access control requirements.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; D.2.6 [**Software Engineering**]: Programming Environments

General Terms

Design, Experimentation, Languages, Security

Keywords

Context-Aware Computing, Pervasive Computing, Context-based Access Control, RBAC

*This work was supported by NSF grants 0411961, 0708604.

SACMAT'08, June 11–13, 2008, Estes Park, Colorado, USA.

1. INTRODUCTION

Context-awareness is a central aspect of pervasive computing applications, characterizing their ability to adapt and perform tasks based on ambient context conditions. There has been steady adoption of context-awareness in number of application domains such as assisted living [8], hospital information systems [5], tour guides [11], and smart environments [30]. At the same time, there has been an increasing concern among researchers for security requirements of such applications [7]. Especially, in the domain of medical information systems such concerns have been recognized for a long time [32, 25], and system models have been developed for access control in such applications [4, 16].

Various definitions of *context* have been proposed in the literature [1, 29]. Broadly, the notion of *context* in pervasive computing applications relates to the characterization of ambient conditions and physical world situations that are relevant for performing appropriate actions in the computing domain for its correct or desired behavior. A person's context is defined in terms of his/her current physical location, devices being used, network on which the devices are connected, and the activities in which the user is currently engaged. Additionally, there can be other conditions and characteristics that may be relevant in defining a context. For example, in some situations the temporal attributes associated with an activity, such as its duration and time of occurrence, may be important. Other factors such as device capabilities, physical proximity of devices, and available bandwidth can also be important in some situations. Some of the typical context-based adaptive characteristics include [29] dynamic integration of resources/services with an application based on context information, context-based access control, displaying information based on the context information, and context-triggered actions.

The focus of this paper is on the issues related to building role-based access control models and systems for pervasive computing applications, utilizing context information in making access control decisions. Several researchers have developed RBAC models that support context-based access control [13, 24, 28, 6, 9, 4, 16, 21]. Context-based constraints may be specified on the User-Assignment (UA) relation [21], or on the activation of role sessions [13], or on the Permission-Assignment (PA) relation [24]. Different kinds of context information have been considered as part of building RBAC systems. These include users' presence in an active space [28], users' presence in specific geographic areas [6], occurrence of specified environmental conditions [9],

a user's memberships in other roles [4] or teams [16], or time intervals as in the GTRBAC model [21].

Integrating context information as part of an access control system is a challenging task due to several reasons. First, acquiring appropriate context information requires interfacing the access control system with various kinds of ambient sensors. Integrity and authenticity of this information is paramount because it may be used in making access control decisions. Second, certain aspects of the context information may be inherently dynamic in nature. During the course of execution of a context-dependent task, it is possible for the related context condition to become false. For certain applications, it may be important that a role member's permissions to execute that task are revoked when such context changes occur. Third, context-based constraints may restrict the resources and services that may be dynamically interfaced with a pervasive computing application.

Pervasive computing applications considered here are deployed in an *active space* containing various kinds of devices, services, and ambient sensors. Context management services are deployed to aggregate sensor data for detecting application-defined context conditions. Resource discovery services are also deployed in an active space. Other active space services register with the discovery services. Applications use the discovery services to dynamically discover and interface with appropriate services in a given context situation. We refer to this as *dynamic object binding*. Applications may define context-based access control policies for such dynamically interfaced services.

The main contribution of this paper is a context-aware RBAC (CA-RBAC) model and its embodiment in a programming framework for designing context-aware applications. The novel features of this programming framework are high-level abstractions for specifying context-based access control requirements, specifically addressing the following aspects:

1. *Role admission and validation constraints:* These constraints specify context-based conditions that need to be satisfied before admitting a user to a role, and also for continuing a user's membership in a role.

2. *Context-based role permissions:* Dynamic object binding causes role operations to interface with different services under different context conditions.

3. *Personalized role permissions:* Such permissions allow different role members to access different active space services based on their individual context.

4. *Context-based permission activation constraints:* These constraints are associated with specific role permissions, and specify context-based conditions that need to hold for a role member to execute such permissions.

5. *Context-based resource access constraints:* These constraints restrict a role member's access to a subset of resources that are managed by an active space service.

The paper is organized as follows. In Section 2 we present two representative context-aware applications which we use to demonstrate the above kinds of access control requirements. In Section 3 we present the CA-RBAC model. In Section 4 we present our programming framework that realizes this model. We compare our work with the related work in Section 5, and conclude in Section 6.

2. RBAC REQUIREMENTS FOR CONTEXT-AWARE COMPUTING

Here we present two representative context-aware applications to motivate the need for extending the NIST RBAC model [13] to support context-based access control requirements.

Context-Aware Patient Information System: Consider a patient information system deployed in a hospital and accessed by the hospital nurses and doctors. It supports a number of different requirements as follows [12]. The system supports assigning a nurse to a ward during certain time periods. During these periods the assigned nurse works in the capacity of a nurse-on-duty role in that ward. In this role the system permits access to the records of only those patients who are admitted to the ward where the nurse is currently present. The nurse's membership in the nurse-on-duty role is revoked when she leaves the ward, or after the end of her duty time. The system permits doctors to create different kinds of reports about patients. For a nurse, access to doctor's reports is allowed only if some doctor is present in the ward where the nurse is located. It may happen that a nurse initiates access to the doctors' reports while a doctor is present in the ward, but the doctor leaves the ward afterwards. In such a case the nurse's on-going session accessing such reports needs to be terminated. This ensures that a nurse does not continue to access these reports in the absence of a doctor. This system may also support location-based access to active space services by nurses and doctors.

Context-Aware Music Player: Consider a music player application which runs on a user's mobile device. Depending on the user's physical location, it streams music either to the user's device or to the audio player service in the room where the user is currently present. We consider the following context-based access control requirements for this application. When the user enters a room, the access control system should allow the application to automatically start streaming music to that room's audio player service if no other user is present in the room. The access control system must revoke the application's access to the room's audio player service when the user leaves the room or some other person enters the room.

2.1 Role Model

Role-based models such as the NIST RBAC have been traditionally used for designing access control systems for organizations. In such systems, roles generally have a long lifetime. Users are assigned to a role by the system administrator, and such memberships also tend to have long duration. In contrast to this, in our model roles are defined as part of an application's design. Such roles come into existence only when that application is deployed and executed, and they last only during the application's lifetime. In our model, we use the following RBAC terminology [2]. Users may be *added* to the roles at the time of application deployment, or they may request to *join* a role when the application is executing. The access control system underlying the application execution environment *admits* a user to a role based on the role admission constraints. A user admitted to a role is called

a *role member*. The permissions associated with a role are represented by a set of operations through which a role member may access an *object*. The execution of such operations by a role member amounts to role *activation*. In this model, there is no explicit notion of *sessions* as in the NIST model. This is because the static and dynamic separation-of-duty constraints can be specified using *event history* based constraints on role operation executions [2]. In some applications, a user's membership in a role may be transient, as it may need to be revoked depending on the context conditions.

2.2 Role admission and validation

User admission to a role may be based on different kinds of context information such as temporal constraints, prior membership in other roles, or user location. Examples of each of these constraints are seen in the patient information system. In this application we may define different roles such as *Nurse* and *Doctor*. We may also define a *NurseOnDuty* role for different wards. Temporal constraints may be used to restrict user membership in the *NurseOnDuty* role only during certain time periods. Prior role membership constraints may be used to restrict only the members of the *Nurse* role to be admitted to the *NurseOnDuty* role for a particular ward. A role member's physical location may be used to allow only those nurses who are physically present in a ward to be admitted to the *NurseOnDuty* role for that ward.

A complementary aspect to context-based role admission is the need to revoke a user's role memberships when specified context conditions fail to hold. For example, in the patient information system a nurse's membership in the *NurseOnDuty* role needs to be revoked when the nurse leaves that ward or after expiration of the specified time interval. We call the above requirement as *role validation constraint*. It determines the validity of a user's membership in a role.

In the NIST RBAC model, there is no explicit notion of role membership revocations. Role membership revocation requirements have been considered by the OASIS RBAC model [4], where a user's membership in a role may be contingent on the membership in some other roles. Some models [21, 6] have used the notion of deactivating a role under certain context conditions. A deactivated role becomes inaccessible to all of its members. In our model, a specific user's privileges for a role are revoked by removing the user from the role membership. Such role membership revocation can be crucial for enforcing policies that may be sensitive to role membership cardinalities.

2.3 Context-based role permissions

In the NIST RBAC model the set of objects for which access control needs to be enforced are statically known. However, in pervasive computing applications specific services that may be accessed by a role member may not be known a priori. They may depend on the context information associated with an application. The application may need to *discover* an appropriate service in the active space and grant access to it under certain context conditions. Because the specific services with which the application would be interfaced is generally not known a priori, the permissions may only be specified on an abstract object. At runtime, this object is dynamically bound to a specific service available in the active space.

We see above kind of requirement in the context-aware music player application. We need the music to stream either to the user's device or to the audio player service in the room where the user is present. Moreover, we need the music to stream to the room's audio player service only if no other person is present in the room. To program this requirement we may define an abstract object named *audio player* to represent the service through which the music would be played. This object would be dynamically bound either to the audio player service in the room where the user is present, or to the audio player service on the user's device. We may also define a *User* role and provide the permission to *play* music on the *audio player* object. As part of binding the object we first authenticate the audio player service and then check for its compatibility to allow invocation of the permission to play music.

2.4 Personalized role permissions

The services that are accessible through a role permission may be different for different role members and may depend on the context information associated with a role member. In the NIST RBAC model the set of objects accessed through a permission are always the same for all the members of a role. A role permission invoked by any member of a role is executed on the *same* object. However, this model is inadequate for pervasive computing applications where a permission invoked by *different* role members may need to be invoked on *different* object instances based on each role member's individual context, such as the physical location.

As an example consider the permission to *print* which is associated with the *Nurse* role in the patient information system. We may require that when this permission is invoked by a nurse, the system chooses the most appropriate printer for satisfying that request. The choice of a particular printer for a specific nurse may depend on various things such as the nurse's preference for a specific printer, context information such as the nurse's location, or the physical security of the printer, or the security level associated with the material being printed.

2.5 Context-based permission activation

Execution of role operations by role members may need to be constrained based on context conditions. The access control model needs to support specification of context-based constraints that need to be satisfied *before* a role member may execute an operation.

Certain operation executions may lead to interactive sessions of arbitrary duration with one or more objects. In some applications we may need such sessions to remain active only under certain context conditions. Correspondingly, the access control model needs to provide mechanisms to terminate sessions initiated through the operation execution when the corresponding context conditions fail to hold. We refer to such changes in required context conditions as *context invalidations*.

We see both the above requirements in the patient information system and in the music player application. In the patient information system we need to restrict a nurse's access to patient reports only if the nurse is co-located with a doctor in a hospital ward. This requirement can be modeled as a context-based constraint on operation execution by nurses. The invocation of the operation by a nurse to access patient reports may lead to initiation of a session with the

database service. We may require that this session should be terminated when either the nurse or the doctor leaves the ward. This is an example of context invalidation.

In the music player application we require that the access control system should grant access for a room's audio player service to a user only if that user and no other person is present in that room. This can be modeled as a context-based constraint on permission to play the music by the application role. The context condition corresponding to the user being alone in a room is invalidated when some other person enters the room while the music is being played on the room's audio player service. In this case the access control system needs to revoke the application's access to the room's audio player service. Moreover, this access also needs to be revoked when the user leaves the room.

We observe that currently such context-based permission activations and revocations are not supported in the NIST RBAC model. The following mechanisms are needed for this purpose. First, we need mechanisms to integrate context information as part of constraint specification on permission activation in a RBAC model. Second, we need mechanisms to continuously monitor context conditions that need to hold while a permission session is active. Third, we need revocation mechanisms to terminate such sessions when the corresponding context conditions fail to hold.

2.6 Context-based resource access

There is a need to distinguish between a service and a resource for access control purpose. A service may be managing a number of resources of a specific type. We may need to control a role member's access to a subset of these resources based on context conditions. For example, in the patient information system the *database service* controls access to the database tables and determines who gets access to them and under what conditions. Such a database may store information about patients such as doctor reports, prescriptions, tests performed, last checkup time, and patient's ward. In this application we require that a nurse should be able to access records of only those patients who are admitted to the ward where the nurse is currently located. This may be satisfied by requiring that the database service grants access for a patient's record to a *Nurse* role member only if that nurse is currently present in the patient's ward. We call such constraints *resource access constraints*. In the literature, similar requirements have been identified and addressed as part of the role graph model [15]. The *parameterized roles* in that model are similar to the *resource access constraints* in our model.

We observe that the role permission activation mechanism presented in Section 2.5 is inadequate for specifying such constraints requiring fine-grained access control of resources managed by a service. This is because of the following reasons. First, the specific resource to which access needs to be granted may not be known a priori. The resource is only identified at runtime, based on dynamic context information. Second, using permission activation to enforce such access control requirement may also lead to information leakage. A role member would be able to find out information about a resource's attributes simply through the failure or the success of the access attempt without ever requiring to access the resource.

3. CONTEXT-AWARE RBAC MODEL

In Figure 1 we present the elements of the CA-RBAC model. We distinguish between the context management layer and the access control layer.

3.1 Context management layer

Expressiveness of the context-based constraint specifications as part of the access control layer depends on the context models that are defined by the application. It is the responsibility of the application designer to define the appropriate context models based on the application requirements. Design of such models also depends on the available sensing technologies. For example, a nurse's location may be modeled at the granularity of a ward or based on the proximity of the nurse to a specific patient in a ward. The available location tracking sensors would determine this granularity.

The purpose of context models is to drive the design of the context management services for aggregating sensor data to generate context information required by an application. In the literature different approaches have been used for context modeling [31]: These include attribute-value pairs to represent context elements [29], domain specific ontologies, such as RDF and OWL [35], and the graphical approach involving Object-Role Modeling (ORM) framework [18, 19]. Attribute-value pairs are easy to handle but provide limited functionality. XML schemas support interoperability and shared understanding of a context model among different consumers of the context information. Ontologies provide mechanisms to define relationships between different context elements and domain concepts. The ORM approach [19] provides graphical mechanisms that aid in the development of context models for the domain of interest.

Ambient context detection requires continuous sensing of various different kinds of conditions in the environment, possibly at different locations, and real-time aggregation of the continuous streams of sensor data to infer the context conditions of interest. Application specific processing of this sensor data may be required to derive high-level context information. Applications may create and install one or more *context agents* for this purpose. The context agents need to authenticate the sensors from which they would collect sensor information. An application needs to trust the context agents from which it obtains the required context information because it is used in making access control decisions. Similar to our notion of context agents other researchers have developed different programming abstractions for accessing context information. These include context widgets [27], situations [19], and sentient objects [14].

In addition to ambient context, an application may also utilize *internal* context information in making access control decisions. Internal context corresponds to role membership related information and history of operation executions by role members. The role managers and the object managers provide interfaces to access this information.

In our model, context conditions are expressed as *predicates* that are evaluated by role/object managers. Such predicates are composed of queries supported by context agents and role managers. These predicates are used by the role/object managers as part of context-based access control policy enforcement. Occurrence of certain context conditions is communicated by the context agents to the access control layer through the notification of context events.

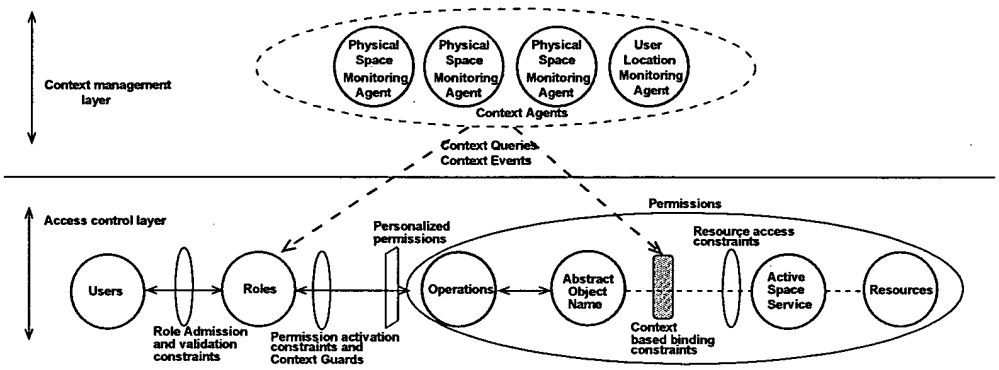


Figure 1: Context-Aware RBAC Model (CA-RBAC)

3.2 Access control layer

One major distinction between our CA-RBAC model and the NIST RBAC model lies in the concept of *permissions* as considered in the two models. In the NIST RBAC model a permission specifies approval to perform specified actions on a particular object. The effect of a permission execution by different role members is always the same; it does not differ for different role members. Also, the objects on which permissions are specified are *statically* known in the system.

Our definition of a *permission* differs from that of the NIST RBAC model as follows. First, permissions may be *personalized* for each role member. A role operation when executed by different role member may access different objects based on the context of the individual role members. Second, permissions are specified on *objects* that may bind to different active space services under different context conditions.

To support *personalized permissions*, objects in our model are classified as *shared* and *private*. Shared objects are common to all members of all the roles, whereas private objects are specific to a role and are managed separately for different members of that role. The bindings of both shared and private objects may change based on context conditions. Personalized permissions are specified as role operations on the private objects defined in a role. The objects accessed by a role member through the personalized permissions belong to that member's private object space.

A *resource access constraint* can be associated with a role operation to enforce context-based access to resources managed by the service that is being accessed as part of that role operation. Such a constraint is a context-based predicate specifying a relation between a resource's attributes and the context variables associated with the role member invoking the operation. Only those resources that satisfy this predicate are allowed to be accessed through the role operation. The object manager that binds to the service evaluates these constraints in its interactions with that service. As part of evaluating these constraints, it utilizes the context information associated with the role member invoking the role operation.

To support context-based permission activation, the model provides the *precondition* mechanism. A precondition can be associated with a role operation. It consists of predicates involving queries to context agents, and operation event count based predicates [2]. It is evaluated by the role manager before the corresponding operation is allowed to be executed.

For addressing the *context invalidation problem*, we provide the *context guard* mechanism in our model. A context guard may be associated with each role operation individually. It consists of a context predicate that is required to remain valid while a role member's interactive session with the service that is being accessed through that role operation is active. The role manager evaluates the context guard and terminates the interactive session if the context predicate fails to hold.

4. PROGRAMMING FRAMEWORK

Based on the CA-RBAC model we have designed and implemented an XML-based programming framework for building context-aware pervasive computing applications [33]. In this framework, a context-aware application is programmed using an abstraction called *activity*. An activity may be distributed across different active spaces and it may involve multiple users in some collaborative tasks.

An activity provides the *role* abstraction. Objects that are defined in the activity are *shared* by all the members of all the roles, whereas objects that are defined within each role are *private* to that role. An object can be specified to bind to some service in the active space or to some context agent. Within an activity we do not distinguish between objects that refer to context agents and those referring to other services. This allows us to treat all the objects in a uniform way. However, there are two consequences of this decision. First, within an activity we need to specify an *order* for binding the objects. It is crucial to bind the objects referring to context agents *before* binding other objects in the activity for an application's correct behavior. In the programming framework we provide a construct for this purpose. Second, based on a given specification, the underlying middleware needs to grant permissions to the various role managers for accessing the context agents as part of role operation precondition evaluations.

The middleware provides three generic components, an *activity manager*, a *role manager*, and an *object manager*. The runtime environment of an application is constructed by specializing these managers based on the application's specification. All the managers are executed on a set of trusted servers in the active space. For each user, an interface component called *User Coordination Interface*

(UCI) is dynamically created and transported to that user's device. Through the UCI a user contacts a role manager for executing a role operation.

A separate object manager is created for each object defined in the activity. Similarly, for objects defined within a role, separate object managers are created corresponding to each user admitted to the role. An object manager maintains a reference to the service to which it is currently bound. It enforces context-based binding policies. It also enforces the resource access constraints that are supplied to it by the role manager as part of role operation invocation.

A separate role manager is created for each role defined in the activity. It enforces role admission and validation constraints, operation preconditions, and context guards. Role validation constraints and operation preconditions are evaluated every time a user invokes a role operation. An operation's actions are performed only if the operation precondition and the role validation constraints are satisfied.

As part of an operation execution, the role manager contacts the object manager corresponding to the object on which the operation is specified to be executed. The object manager invokes on the currently bound service the methods that are specified as part of the role operation. For monitoring and gathering external context information we use an agent-based distributed event monitoring system [34].

We now illustrate our programming framework by presenting how access control requirements for the two applications presented in Section 2 can be programmed in our framework. For both the applications we used the following experimental setup. We defined *room agents* corresponding to various rooms in our department building. Each such agent maintained status information about its room, such as the list and the number of users in the room. Users' Bluetooth enabled devices, such as laptops, and PDAs, are used to detect user presence in a room. The agent generated the *StatusChangeEvent* when a person arrived or left the room. A user location monitoring agent was defined in the testbed environment to maintain location information for each user. In the activity we defined a *LocationServiceAgent* object to refer to this agent. This agent subscribed to the *StatusChangeEvent* from each room agent in the system. It generated *LocationChangeEvents* corresponding to each user as he/she moved from one room to another.

4.1 Role admission and validation

The programming framework supports two constructs for programming role admission and validation constraints that are specified with each role. The specified constraints are evaluated by the corresponding role manager. Validation constraints are evaluated every time a role member attempts to execute a role operation. Below we illustrate the usage of these constructs as part of the patient information activity.

We consider the following two requirements related to a user's memberships in a nurse-on-duty role associated with a ward. First, we require that a user should be admitted to the nurse-on-duty role corresponding to a ward only if he/she is also a member of the *Nurse* role. Second, we require that the user's membership in the nurse-on-duty role of a ward should be revoked if the user goes out of that ward, or after his/her time of duty is over, or if the user's membership in the *Nurse* role is revoked.

Below we present the partial specification of the *PatientInformationSystem* activity, enforcing these two requirements. We use a pseudo notation for presenting the specification examples. In this notation the **boldface** terms represent XML tags in our programming framework. We define the *NurseOnDutyEmergencyWard* role and the *EmergencyWardAgent* object in this activity. We bind the object to a specific ward's context agent at the activity instantiation time.

```
Activity PatientInformationSystem {
   Role NurseOnDutyEmergencyWard {
      AdmissionConstraint { member(thisUser, Nurse) }
      ValidationConstraint {
         EmergencyWardAgent.isPresent(thisUser) &&
         current_time <= DATE (Mar, 21, 2008, 10:00) &&
         member(thisUser, Nurse)
      }
   } // end of role
   Object EmergencyWardAgent {
      Bind Direct (//WardAgentURL)
   } // end of object
}
```

The first requirement is specified through the *AdmissionConstraint* construct. As part of this constraint we check for the user's membership in the *Nurse* role. The function *member(thisUser, RoleName)* is provided in the programming framework to check the invoking users membership in the role *RoleName*. The variable *thisUser* is a special variable in the programming framework which translates to the identifier of the role member who is requesting admission to the role.

The second requirement is specified through the *ValidationConstraint* construct. As part of this constraint the role manager queries the *EmergencyWardAgent* object to check whether this role member is present in the ward. If the role member is not present in the ward, or if the current time is past the specified time, or if the user is no longer a member of the *Nurse* role then the user's membership in the *NurseOnDutyEmergencyWard* role is revoked.

4.2 Context-based object bindings

A context-based permission is programmed as a role operation on an object which may bind to different active space services under different context conditions. For programming an object's context-based binding policies, the *Reaction* construct is provided in the programming framework. One or more reactions may be specified with an object definition. All such reactions are handled by the corresponding object manager. A reaction follows the Event-Condition-Action (ECA) model of evaluation. It is triggered by one or more *context events*. The condition specification consists of queries to context agents. The action consists of the binding specification for the object.

In order to illustrate the context-based object binding and the use of the *Reaction* construct, we consider the context-based requirements of the music player application. In this application we require that when a user enters a room, the application is able to access the audio player service in that room only if no other user is present in the room. In Figure 2 we present a partial specification of this activity that only addresses the above requirement. Other requirements presented in Section 2 for this application are not considered here.

We define two objects, *CurrentRoom* and *AudioPlayer*, within the *User* role. Both these are *private* objects of this role. Through the *BindingOrder* construct we specify that

```
Role User {
   BindingOrder { CurrentRoom AudioPlayer }
   Object CurrentRoom RDD (//RoomRDD.xml) {
      Reaction {
         When Event LocationChangeEvent(thisUser)
            Bind Discover
            (LOCATION=LocationServiceAgent.getLocation
               (thisUser))
      } // end of binding reaction
   } // end of object definition
   Object AudioPlayer RDD (//AudioPlayerRDD.xml) {
      Reaction {
         When Event LocationChangeEvent(thisUser)
         Precondition CurrentRoom.isPresent(thisUser) &&
            CurrentRoom.presentUserCount() == 1
            Bind Discover
            (LOCATION=LocationServiceAgent.getLocation
               (thisUser))
      } // end of reaction
   } // end of object definition
   Operation PlayMusic {
      Precondition AudioPlayer.isBound()
         Action AudioPlayer.play()
   } // end of operation
} // end of User Role
```

Figure 2: Context-based object binding example

the *CurrentRoom* object should be bound *before* binding the *AudioPlayer* object. This is crucial for the correct execution of this activity because the *CurrentRoom* object is accessed as part of the condition evaluation of the *AudioPlayer* object's binding reaction.

The binding reaction of the *CurrentRoom* object is triggered by the *LocationChangeEvent* corresponding to the user. The object is bound by *discovering* the room agent based on the role member's location. Such a discovery is performed as follows. Associated with each object is a XML description called RDD (Resource Description Definition), which specifies the attributes and interfaces of the service to which that object may be bound. We use RDD for service discovery in our system. Some of the attributes' values in a RDD can be based on the context information at the discovery time. In the above example, the *LOCATION* attribute value in the *RoomRDD* is filled in at runtime by querying the location of the role member from the *LocationServiceAgent*.

For the *AudioPlayer* object we specify a reaction that binds the object to the audio player service in the room where the user is present. This reaction is triggered by the *LocationChangeEvent* corresponding to the user. As part of the reaction's condition, the *AudioPlayer* object manager queries the room agent to which the *CurrentRoom* object is bound. The condition checks whether this role member is the only person present in the room. The binding action is performed if this condition is true. The discovery procedure is same as the binding of the *CurrentRoom* object. The user may initiate playing of music through the *PlayMusic* operation.

4.3 Personalized role permissions

A personalized role permission is programmed as a role operation on a *private* object defined in a role. We illustrate this through the following example. In the patient information system we want that every member of the *Nurse* role should be able to access the printer service in the ward where that member is located. In Figure 3 we show how this requirement is programmed in our framework. In the *Nurse* role we define the *Print* operation through which the private *MyPrinter* object may be accessed.

```
Role Nurse {
   Object MyPrinter RDD (//PrinterRDD.xml) {
      Reaction { ... }
   } // end of MyPrinter object definition
   Operation Print {
      Action MyPrinter SessionMethod print
   } // end of operation
} // end of Nurse Role
```

Figure 3: Personalized role permissions

The *MyPrinter* object refers to separate printer service instances for each role member. Therefore, the binding of this object is maintained separately and independently for each *Nurse* role member. The binding action for this object is similar to the binding of the *CurrentRoom* object presented in Figure 2.

In the programming framework the *SessionMethod* construct is provided to specify the list of methods that are allowed to be invoked as part of an *interactive session*. We observe that such an interactive session is used only for a role member's interaction with a service. It should not be confused with the concept of a *session* in the NIST RBAC model.

4.4 Permission activation and context guard

The context-based permission activation constraints are programmed as a *precondition* of a role operation. A role manager evaluates an operation's precondition before the operation's actions can be executed.

```
Role Nurse {
   Object CurrentWard { ... }
   Operation AccessCriticalReports {
      Precondition
         CurrentWard.isPresent(thisUser) &&
         CurrentWard.isPresent(members(Doctor))
      Action PatientDB
         SessionMethod accessDoctorReport
      ContextGuard {
         When StatusChangeEvent
         GuardCondition
            CurrentWard.isPresent(thisUser) &&
            CurrentWard.isPresent(members(Doctor))
      } // end of Context Guard
   } // end of role operation
} // end of Nurse role
```

Figure 4: Operation precondition and context guard example

In the patient information system we require that a *Nurse* role member may access doctor's patient reports only if a *Doctor* role member is also present in the ward. This requirement is programmed using the precondition construct as shown in Figure 4. We define the private object named *CurrentWard* in the *Nurse* role. For each nurse it is bound to the room agent corresponding to the room where that nurse is present. The object's binding changes based on the

current location of a nurse. In this regard it is similar to the binding of the *CurrentRoom* object presented in Figure 2.

We define the *AccessCriticalReports* operation through which a *Nurse* role member may access doctor reports. As part of the operation precondition the *Nurse* role manager checks if the *Nurse* role member who is invoking the operation and some member of the *Doctor* role are present in the *CurrentWard*. The role manager executes the operation's actions only if the precondition is true. Execution of this operation leads to the initiation of an interactive session with the *database service*. As part of this session the role member may invoke *accessDoctorReport* method on the *PatientDB* object. We want that this session be terminated when either the nurse leaves the ward or when no member of the *Doctor* role remains in the ward.

In our programming framework we provide the *ContextGuard* construct for addressing the above kind of context invalidation problem. Such a construct identifies two things - *what* context predicate to evaluate, and *when* to evaluate it. In our framework, we use *context events* to trigger the evaluation of a context guard. A context guard becomes effective when the corresponding operation is executed. Once effective, the role manager evaluates the context condition specified through the *GuardCondition* construct every time the guard's evaluation is triggered by the notification of the specified context events. The role manager terminates the interactive session if the context condition fails to hold.

We specify a context guard for the *AccessCriticalReports* operation. Its evaluation is triggered by the *StatusChangeEvent* that is notified to the *Nurse* role manager by the ward agent to which the *CurrentWard* object is bound. The *Nurse* role manager terminates the interactive session if either the nurse has left the ward or if no member of the *Doctor* role is present in the ward.

4.5 Resource access constraint

A resource access constraint may be specified separately for every object invocation within a role operation. We define the *AccessConstraint* construct for this purpose in the programming framework.

```
Role Nurse {
   Operation AccessWardPatientInfo {
      Action PatientDB
         SessionMethod accessPatientInformation
         AccessConstraint
            (WardID =
            LocationServiceAgent.getLocation(thisUser))
   }
} // end of Nurse role
```

Figure 5: Resource access constraint example

In the patient information system we require that a *Nurse* role member may access the records of only those patients who are admitted to the ward where the nurse is currently present. This requirement is programmed as shown in Figure 5. In the *Nurse* role we define the *AccessWardPatientInfo* operation through which a role member may access patient records. Through the *AccessConstraint* construct we specify the required resource access constraint. It is enforced by the *PatientDB* object manager. The constraint specification consists of the *WardID* attribute of patient records. The *PatientDB* object manager grants access to only those database records for which the *WardID* attribute has the value equal to the location of the *Nurse* role member who is invoking the operation. For this, it queries the *LocationServiceAgent* to obtain the location of the nurse.

5. RELATED WORK

Other researchers have also developed RBAC models specifically for context-aware pervasive computing applications [28, 9, 24]. Gaia [28] defines three different role categories, corresponding to system-wide roles, active space roles, and application roles, and a mapping between them. In GRBAC model [9] context information is considered as the *environmental role*, which an application needs to posses in order to perform context-dependent tasks. Such a definition leads to large number of roles in an access control system, as there might be potentially many environmental states that are relevant for an application. In [24], context-based constraints are associated with activation of role permissions. They also provide engineering guidelines for building context-based RBAC systems.

The context-based access control mechanisms in our programming framework differ from the above systems in the following four ways. First, we provide role validation constraints to support revocations of role membership if context conditions fail to hold. Second, we support personalized role permissions that are executed on different objects for different role members. Third, we provide the *context guard* mechanism to revoke interactive sessions when context conditions fail to hold. Fourth, through the *resource access constraint* mechanism we enforce access control requirements that depend on the relationship of a resource's attributes with the context information, such as the identity and the location of the role member who is accessing the resource. Similar requirements have been discussed and addressed previously in [17, 15]. The mechanisms of *parameterized privileges* [17] and *parameterized roles* [15] essentially perform access control based on an object's contents. Content-based access control ideas can be traced back to the work on access control based on *data types* in programming languages [20].

Constraints specification as part of RBAC models have been extensively studied by researchers [3, 10]. We focus on context-based constraint specification as part of designing context-aware applications. This requires close interaction between the access control system and the context management layer. In our model, application specific context agents are deployed in the system to collect context information and generate context events. The context predicates are evaluated by the application specific role/object managers. A generic framework for context evaluation has been developed as part the Antigone system [22]. It provides mechanisms to enforce security of the context information used as part of authorization policies. In contrast, our work considers integration of context-based constraints in a RBAC model for pervasive computing applications. This leads to a number of requirements, discussed in this paper, which are not addressed by the Antigone framework.

Context-based constraints that limit a role's visibility to specific geographic areas are presented as part of the GEO-RBAC model [6]. Similarly, the GTRBAC model [21] provides mechanisms for enabling and disabling of roles based on temporal constraints. In our model context-based

constraints including temporal and spatial constraints, are specified as part of role admission/validation and role operation preconditions. We argue that this approach supports fine-grained access control requirements, as one can selectively revoke a user's membership from a role, or activate/deactivate specific role permissions, instead of enabling/disabling a role. Moreover, we also support fine-grained access control through *resource access constraints* that are specified as part of a role operation. Additionally, we also provide the context guard mechanism to revoke role operation sessions when specified context conditions fail to hold.

The UCON model [26] provides an extensive framework to model a broad range of *usage control* policies. In contrast, we focus on access control for context-aware applications. There are conceptual similarities between some of the mechanisms in our programming framework and some of the UCON modeling abstractions. For instance, the context guard mechanism in our framework is similar to the UCON's decision predicates that are evaluated during a request execution (*ongoing-authorizations*). Moreover, the UCON's attribute-based access control mechanisms can be used to specify the requirements that are addressed by the *resource access constraints* in our framework. Support for attribute-based access control is also provided by the XACML standard for distributed authorization management [23].

The *resource access constraint* mechanism in our programming framework is similar to the *parameterized roles* of the role graph model [15] and *parameterized privileges* of [17]. Additionally, we also provide the mechanism of *personalized role permissions* in our model. The personalized role permissions and the resource access constraint mechanism serve different purposes. Personalized role permissions (private object space for a role member) provide a mechanism to control a role member's access to a service based on his/her context information. Resource access constraints on the other hand provide a mechanism to control a role member's access to a subset of resources managed by a service. A resource access constraint provides a finer granularity of access control as compared to the personalized role permissions.

Distinction between an object type and its instance for access control has also been considered in the TMAC model [32]. In contrast to that model, the abstract objects in our model may be dynamically bound to different services under different context conditions. This requires object managers to verify the authenticity of such services before binding. Such a requirement does not arise in the TMAC model, making its implementation possibly simpler than our model.

6. CONCLUSIONS

In this paper we have demonstrated the need for extending the NIST RBAC model for addressing context-based access control requirements of pervasive computing applications. We presented our context-aware RBAC model that supports several such extensions. In this model we distinguish between the context management layer and the access control layer. The model supports personalized permissions for role members, and context-based constraint specification as part of - dynamic binding of objects with active space services, user admission to roles, permission executions by role members, and granting access to a subset of a service's resources based on a role member's context information. The model also supports revocation of a user's membership in a role when context conditions fail to hold. We have also identified the *context invalidation problem* in this paper. To address this problem we provide the context guard mechanism in our CA-RBAC model. Based on this model we have developed a role-based framework for programming secure context-aware pervasive computing applications. In this paper we demonstrated the key elements of this framework through a set of application examples that we have implemented as part of our testbed suite.

7. ACKNOWLEDGMENTS

We are thankful to Prof. Sylvia Osborn and all the reviewers for providing valuable feedback that resulted in improving the paper's presentation.

8. REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles. Towards a Better Understanding of Context and Context-Awareness. In *HUC '99: Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, pages 304–307. Springer-Verlag, 1999.

[2] T. Ahmed and A. R. Tripathi. Specification and Verification of Security Requirements in a Programming Model for Decentralized CSCW Systems. *ACM Transactions on Information and System Security (TISSEC)*, 10(2):7, 2007.

[3] G.-J. Ahn and R. Sandhu. Role-based Authorization Constraints Specification. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):207 – 226, November 2000.

[4] J. Bacon, K. Moody, and W. Yao. A Model of OASIS Role-based Access Control and its support for Active Security. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):492–540, 2002.

[5] J. E. Bardram, T. R. Hansen, M. Mogensen, and M. Søgaard. Experiences from Real-World Deployment of Context-Aware Technologies in a Hospital Environment. In *Ubicomp*, pages 369–386, 2006.

[6] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. GEO-RBAC: A Spatially Aware RBAC. In *SACMAT '05: Proceedings of the Tenth ACM Symposium on Access control Models and Technologies*, pages 29–37, 2005.

[7] R. Campbell, J. Al-Muhtadi, P. Naldurg, G. Sampemane, and M. D. Mickunas. Towards Security and Privacy for Pervasive Computing. In *Lecture Notes in Computer Science Software Security - Theories and Systems*, volume 2609, pages 77–82. Springer, 2003.

[8] S. Consolvo, P. Roessler, B. E. Shelton, A. LaMarca, B. Schilit, and S. Bly. Technology for Care Networks of Elders. *IEEE Pervasive Computing*, 3(2):22–29, 2004.

[9] M. J. Covington, W. Long, S. Srinivasan, A. K. Dey, M. Ahamad, and G. D. Abowd. Securing Context-Aware Applications Using Environment Roles. In *SACMAT '01: Proceedings of the Sixth ACM Symposium on Access control Models and Technologies*, pages 10–20, 2001.

[10] J. Crampton. Specifying and Enforcing Constraints in Role-based Access Control. In *SACMAT '03: Proceedings of the Eighth ACM Symposium on Access control Models and Technologies*, pages 43–50, 2003.

[11] N. Davies, K. Cheverst, K. Mitchell, and A. Efrat. Using and Determining Location in a Context-sensitive Tour Guide. *IEEE Computer*, 34(8):35–41, August 2001.

[12] M. Evered and S. Bögeholz. A Case Study in Access Control Requirements for a Health Information System. In *ACSW Frontiers '04: Proceedings of the Second Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 53–61, 2004.

[13] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for Role-based Access Control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[14] A. Fitzpatrick, G. Biegel, S. Clarke, and V. Cahill. Towards a Sentient Object Model. In *Workshop on Engineering Context-Aware Object-Oriented Systems and Environments (ECOOSE)*, November 2002.

[15] M. Ge and S. L. Osborn. A Design for Parameterized Roles. In *DBSec*, pages 251–264, 2004.

[16] C. K. Georgiadis, I. Mavridis, G. Pangalos, and R. K. Thomas. Flexible Team-based Access Control using Contexts. In *SACMAT '01: Proceedings of the Sixth ACM Symposium on Access control Models and Technologies*, pages 21–27, 2001.

[17] L. Giuri and P. Iglio. Role Templates for Content-based Access Control. In *RBAC '97: Proceedings of the Second ACM Workshop on Role Based Access Control*, pages 153–159, 1997.

[18] T. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design.* Morgan Kaufmann Publishers Inc., 2001.

[19] K. Henricksen and J. Indulska. A Software Engineering Framework for Context-Aware Pervasive Computing. In *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, page 77, 2004.

[20] A. K. Jones and B. H. Liskov. A Language Extension for Expressing Constraints on Data Access. *Commun. ACM*, 21(5):358–367, 1978.

[21] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A Generalized Temporal Role-Based Access Control Model. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)*, 17(1):4–23, 2005.

[22] P. McDaniel. On Context in Authorization Policy. In *SACMAT '03: Proceedings of the Eighth ACM Symposium on Access control Models and Technologies*, pages 80–89, 2003.

[23] T. Moses. OASIS eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard. pages 1–141, 1 February 2005.

[24] G. Neumann and M. Strembeck. An Approach to Engineer and Enforce Context Constraints in an RBAC Environment. In *SACMAT '03: Proceedings of the Eighth ACM Symposium on Access control Models and Technologies*, pages 65–79, 2003.

[25] U. Nitsche, R. Holbein, O. Morger, and S. Teufel. Realization of a Context-Dependent Access Control Mechanism on a Commercial Platform. In *Proceedings of IFIP/SEC 1998.* Chapman & Hall.

[26] J. Park and R. Sandhu. The $UCON_{ABC}$ Usage Control Model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):128–174, 2004.

[27] D. Salber, A. K. Dey, and G. D. Abowd. The Context Toolkit: Aiding the Development of Context-Enabled Applications. In *Proceedings of the 1999 Conference on Human Factors in Computing Systems (CHI '99)*, pages 434–441, May 1999.

[28] G. Sampemane, P. Naldurg, and R. H. Campbell. Access control for Active Spaces. In *Annual Computer Security Applications Conference (ACSAC2002)*, 2002.

[29] B. Schilit, N. Adams, and R. Want. Context-Aware Computing Applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, pages 85–90, Santa Cruz, CA, US, 1994.

[30] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao, and F. Liu. The Smart Classroom: Merging Technologies for Seamless Tele-Education. *IEEE Pervasive Computing*, 02(2):47–55, 2003.

[31] T. Strang and C. Linnhoff-Popien. A Context Modeling Survey. In *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing*, September 2004.

[32] R. K. Thomas. Team-based Access Control (TMAC): A Primitive for Applying Role-based Access Controls in Collaborative Environments. In *RBAC '97: Proceedings of the Second ACM Workshop on Role-based Access Control*, pages 13–19, 1997.

[33] A. Tripathi, D. Kulkarni, and T. Ahmed. A Specification Model for Context-Based Collaborative Applications. *Elsevier Journal on Pervasive and Mobile Computing*, 1(1):21 – 42, May-June 2005.

[34] A. R. Tripathi, D. Kulkarni, H. Talkad, M. Koka, S. Karanth, T. Ahmed, and I. Osipkov. Autonomic Configuration and Recovery in a Mobile Agent-based Distributed Event Monitoring System. *Software - Practice & Experience*, 37(5):493–522, 2007.

[35] X. H. Wang, D. Q. Zhang, T. Gu, and H. K. Pung. Ontology Based Context Modeling and Reasoning Using OWL. In *PERCOMW '04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004.

Executable Misuse Cases for Modeling Security Concerns

Jon Whittle
Dept. of Computing
Lancaster University
Lancaster, LA1 4WA
+44 (0) 1524 510492
whittle@comp.lancs.ac.uk

Duminda Wijesekera
Dept. of Information & Software Engineering
George Mason University
Fairfax, VA 22030
+1 703 9931640
dwijesek@gmu.edu

Mark Hartong*
Federal Railroad Administration
1120 Vermont Ave
Washington, DC 20590
+1 202 4936000
mark.hartong@fra.dot.gov

ABSTRACT

Misuse cases are a way of modeling negative requirements, that is, behaviors that should not occur in a system. In particular, they can be used to model attacks on a system as well as the security mechanisms needed to avoid them. However, like use cases, misuse cases describe requirements in a high-level and informal manner. This means that, whilst they are easy to understand, they do not lend themselves to testing or analysis. In this paper, we present an executable misuse case modeling language which allows modelers to specify misuse case scenarios in a formal yet intuitive way and to execute the misuse case model in tandem with a corresponding use case model. Misuse scenarios are given in executable form and mitigations are captured using aspect-oriented modeling. The technique is useful for brainstorming potential attacks and their mitigations. Furthermore, the use of aspects allows mitigations to be maintained separately from the core system model. The paper, supported by a UML-based modeling tool, describes an application to two case studies, providing evidence that the technique can support red-teaming of security requirements for realistic systems.

Categories and Subject Descriptors

D.2.1 **[Software Engineering]**: Requirements/Specifications – *Elicitation methods, Languages*

General Terms

Security, Languages

Keywords

Misuse cases, Scenarios, Early Aspects

1. INTRODUCTION

A misuse case [1, 2] is "a use case from the point of view of an actor hostile to the system under design" [1]. Misuse cases have been used effectively to model potential attacks to a system and to analyze the trade-offs of mitigation strategies [3]. However, misuse cases, just like use cases, are an informal notation and therefore are not directly amenable to formal analysis or testing. This paper presents an executable modeling technique for misuse cases that allows modelers to specify use cases and misuse cases for a system, and then to animate those misuse cases on the use case model. This gives project stakeholders a way to brainstorm potential attacks, model those attacks as misuse cases, model mitigation mechanisms to prevent the attacks, and then animate or test the resulting model to ensure that the attacks are mitigated appropriately. Furthermore, the technique can be used in eliciting new attacks and animating the effects of those attacks on the system behavior. The executable modeling language has a formal definition, yet is based on UML and so remains accessible to stakeholders.

The approach builds upon recent work in software modeling, namely, precise use case modeling [4, 5], aspect-oriented modeling [6-8], and statechart synthesis from scenarios [9]. Use and misuse case scenarios are modeled precisely using extended interaction overview diagrams (EIODs) [5]. Attack mitigation scenarios are modeled as aspects that weave mitigation behavior into the use case model. The entire weaved scenario model can then be transformed automatically into a set of finite state machines (FSMs). These FSMs can be animated or the misuse scenarios can be executed on the FSMs to check that the mitigations do indeed prevent the attacks.

EIODs are a formalization of UML interaction overview diagrams for modeling scenarios precisely. In this paper, we extend EIODs to the modeling of misuse cases by including use/misuse case relationships. Aspect-oriented modeling is a way of separating cross-cutting concerns during software modeling. In this paper, we model attack mitigations as aspect scenarios. This is beneficial because it maintains a clear separation, during requirements modeling, of the core functionality of the system and functionality needed to handle security concerns. The mitigating behavior can be automatically weaved with the use case behavior to obtain a scenario-based description of the complete system. Finally, we use the statechart synthesis algorithm from [9] to make the combined system behavior executable. Statechart synthesis automatically generates a set of communicating FSMs from an EIOD. Since FSMs have a well understood semantics, they can be executed, animated and/or analyzed.

The contribution of this paper, therefore, can be measured in two ways. The first contribution is the integration of these three

ICSE'08, May 10–18, 2008, Leipzig, Germany.

* The views and opinions expressed herein are that of the authors and do not necessarily state or reflect those of the United States Government, the Department of Transportation, or the Federal Railroad Administration, and shall not be used for advertising or product endorsement purposes.

techniques applied to executable misuse case modeling. Secondly, the paper extends two of these three techniques. EIODs are extended to support misuse case modeling and aspect modeling is extended to support expressive weaving strategies for scenarios. The first extension is reasonably straightforward; the second is significant.

The ideas in the paper have been implemented in a tool, MUCSIM (misuse case simulator), which is implemented as a plug-in to IBM Rational Software Modeler. MUCSIM integrates two existing tools, UCSIM [10] and MATA [11]. In particular, MATA was extended to support this work. The techniques have been applied to two realistic case studies. The first is a commercial electronic voting system, previously described by Kohno et al. [12]. The second is a positive train control system (PTC), previously considered by the second and third authors. PTCs are integrated command and control systems for ensuring train safety and are planned for deployment by the Federal Railroad Administration. We modeled security concerns for both applications using our technique. We believe that these two real-world case studies provide evidence for the applicability of the technique presented in the paper.

The paper is organized as follows. Section 2 introduces the key ideas using the electronic voting system as an illustrative example. Section 3 describes the technical contributions of the paper. Section 4 presents detailed results of the two case studies and is followed by related work and conclusions.

2. ILLUSTRATIVE EXAMPLE

Kohno et al [12] describe a security analysis of Diebold's AccuVote-TS Electronic Voting System (EVS). This is a real EVS for which Kohno et al found significant security vulnerabilities. We use the EVS in this paper both as an introductory example but also as a validation of our work. This section presents a small part of the example.

Diebold's EVS works as follows. An EVS allows voters to cast ballots at an electronic voting booth within a polling station. Two high-level business goals for an EVS are: (1) voter anonymity should be maintained, and (2) votes should be captured correctly. There are three principal use cases: (1) ballot definitions (i.e., details of candidates, party affiliations etc.) are sent to all voting machines at all polling stations, (2) voters cast ballots, (3) when voting is complete, the EVS reports the results. Voting is done using a smartcard. A voter registers at a polling station by showing his/her ID and is given a smartcard and PIN. S/he then proceeds to the EVS, enters the smartcard and PIN, and casts a vote.

Kohno et al. [12] list eleven attacks that could be perpetrated on Dielbold's EVS (based on an analysis of its source code). A simple example of a successful attack would be if an attacker were to manufacture his/her own smartcards that could then be used to cast multiple votes for a single voter, thus violating the second business goal above. We show in the remainder of this section how this attack and a possible mitigation can be modeled as executable misuse cases.

The behavior of the EVS must first be modeled as an extended UML interaction overview diagram (EIOD). EIODs will be explained in Section 3.1. We recommend a three level approach

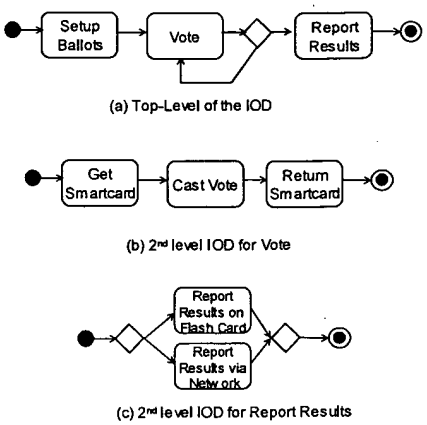


Figure 1: EVS Use Cases as Interaction Overviews.

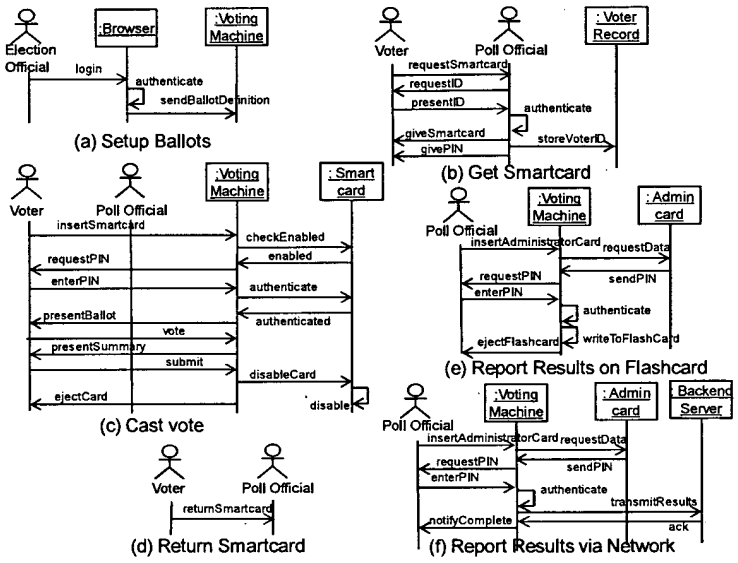


Figure 2: EVS Use Case Sequences.

to modeling with EIODs. The top-level shows the use cases (Figure 1(a)). Each use case is then (optionally) described by another EIOD (Figures 1(b) and 1(c)) and finally, each of these IODs is described by a UML sequence diagram (see Figure 2). The use case model given by Figures 1 and 2 can be input to the UCSIM tool, which will automatically generate a set of FSMs, one for each participant in the sequence diagrams. Animation of these FSMs (available in UCSIM) can be used in validating the behavior of the EVS. FSMs generated for the voting machine, the smartcard and the voter are given in Figure 3.

The attack mentioned above can be captured as a misuse case. In our approach, a misuse case is also modeled by scenarios (i.e., sequences). Figure 4(a) shows how the attack relates to the voting use case. We use standard misuse case relationships. A misuse case <<threatens>> a use case if it potentially could prevent the use case's goal from being achieved. A mitigating case <<mitigates>> a misuse case if it reduces the possibility of the attack being successful.

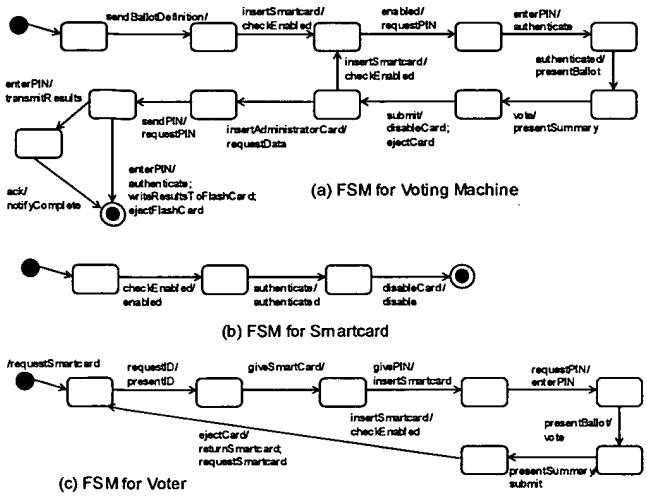


Figure 3: EVS Finite State Machines.

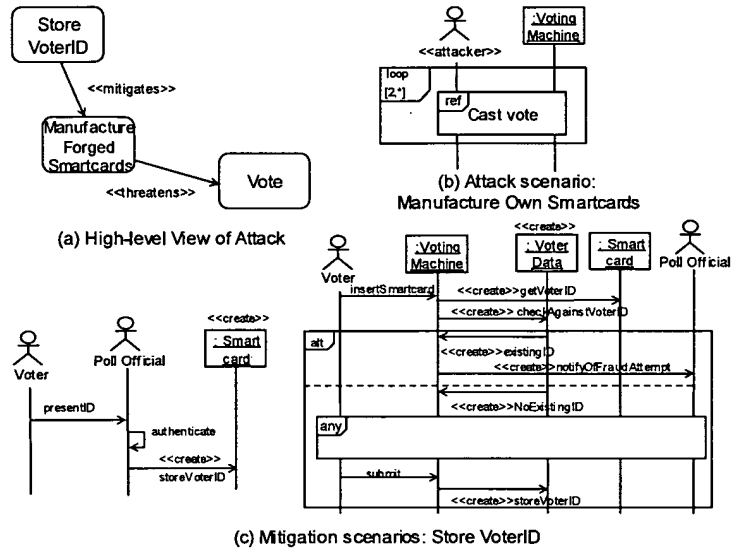


Figure 4: Attack 1 - Manufacture Own Smart Cards.

In order to make the misuse specification executable, the misuse case and mitigating case are described as scenarios. The attack scenario (Figure 4(b)) depicts a sequence of events that, if executed successfully, would constitute a successful attack. In this example, an attacker (stereotyped as <<attacker>>) would be successful if s/he were able to cast two or more votes in succession. This is modeled as a loop referencing the Cast Vote scenario. Note that only the interactions of the attacker with the system are modeled—in particular, it is not of concern how the attacker was able to manufacture additional smart cards.

Based on the current use case model (Figure 1), this attack would be successful because the attack scenario can be executed on the generated FSMs (Figure 3) as follows. The <<attacker>> participant injects messages into the Voting Machine object in the sequence defined by Figure 4(b). In this example, the attack succeeds because there is nothing in the FSMs to stop the attacker from voting twice. In this way, executing the attack scenario is similar to executing a test case on the FSMs.

This attack succeeds because the FSMs do not prevent a second (manufactured) smartcard being inserted to cast an identical second vote. Once the modeler has modeled a successful attack scenario, s/he should design a mitigation strategy. This is done by defining mitigation scenarios—see Figure 4(c). A mitigation scenario is a sequence of events that will prevent the attack or reduce the chances of it being successful. In our approach, a mitigation scenario is an aspect scenario and defines mitigation messages that are weaved automatically into the core behavior. This has the advantage of keeping the mitigations separate from the core behavior so that they can easily be modified later or reused. It also allows stakeholders to easily trade-off different mitigation strategies since it is easy to remove one mitigation strategy and replace it with another.

Figure 4(c) defines two mitigation scenarios for this first attack. In the original definition of the use cases, no voter-specific data was encoded on the voter smartcards. By encoding a unique ID for each voter, the EVS can avoid multiple votes from the same voter by recording the ID when a vote is cast and then checking for previous votes from the same ID when a new vote is cast. Note that only the voter ID, not the vote itself, is stored and so voter anonymity is preserved. Storing a voter ID on the smartcard will thwart a naïve attacker that manufactures multiple faked smart cards but does not encode different IDs on each.

Storing a voter ID on the smartcard is specified on the left side of Figure 4(c). The right side specifies how to check for an existing ID. If one is found, there is a fraud attempt, and so a poll official is notified. The notation used will be explained fully in Section 3.2. For now, it is enough to know that any model element stereotyped with <<create>> will be added to the core behavior. Any element without a stereotype is matched against model elements in the core behavioral model. For example, the scenario on the left side of Figure 4(c) looks for a sequence of messages in the base where *presentID* is followed by *authenticate*. If found, a new message, *storeVoterID*, is added and a new smart card instance is also added. Similarly, on the right side of Figure 4(c), six new messages are added – five after *insertSmartcard* and one after *submit*. A new object, *VoterData*, is also added. The *any* fragment is used to match against a sequence of messages of undefined length. Hence, Figure 4(c) matches against any sequence of messages beginning with *insertSmartcard* and ending with *submit*, i.e., a sequence corresponding to a cast vote.

The MATA tool [11] was extended to automatically weave mitigation scenarios into a set of core behavior scenarios. The UCSIM tool then automatically generates a new set of FSMs that include *both* the original use case behavior and the new mitigation behavior. The attack scenario from Figure 4(b) can then be re-executed on the new set of FSMs to see if the attack still succeeds. In this case, it will not succeed because the check for an existing ID will stop an attempt at a second vote.

To run the attack scenario on the new set of FSMs, the FSM execution tool must be able to interpret the events *existingID* and *NoExistingID*. This is because the new FSM for the voting machine contains a branch after the smart card has been inserted. Two transitions form this branch–one for the case when there is no existing ID and one when there is. To execute the FSM any further, the execution tool must decide which branch

to take. This can be resolved in one of two ways. Either the user directs the execution tool by injecting events. This allows the user to see what happens in both cases. Alternatively, the user can add new message specifications that provide interpretations that the execution tool can use. In this case, the user would add:

```
context storeVoterID
post: stored = true

context existingID
pre: stored = true

context NoExistingID
pre: stored = false
```

There are two points to note about this process. Firstly, executing the attacks and mitigations may lead project stakeholders to realize that whilst they have thwarted this particular attack, a more sophisticated attack (for example, where the attacker generates new IDs for each forged smartcard) may still succeed. This might in turn lead to a new mitigation. Secondly, a new mitigation may allow a previous attack to succeed because it nullifies a previous mitigation strategy. This situation can be easily detected by running all previous attack scenarios as a set of regression tests. Hence, this process can aid stakeholders in iteratively refining mitigation strategies.

3. TECHNICAL CONTRIBUTIONS

This section describes the key contributions of the paper. As stated in Section 1, one contribution is the integration of previous work on aspect-oriented scenarios and FSM synthesis. In addition, EIODs have been extended to support misuse cases (Section 3.1) and we have extended the expressiveness of our scenario weaving mechanism (Section 3.2).

3.1 Misuse Interactions

Extended interaction overview diagrams (EIODs), introduced in [4], are an extended form of UML [13] interaction overview diagram with a formally defined semantics [5], that model use cases at three levels. Level 1 is an extended activity diagram that shows use cases and their relationships. Each level 1 node (i.e., a use case) is refined at level 2 as an extended activity diagram. At level 2, each node is a scenario and the activity diagram therefore shows scenario relationships. Each level 2 node is refined at level 3 as a UML sequence diagram. An EIOD gives a complete, formally defined, description of a set of use case scenarios and the algorithm from [9] can be used to generate executable hierarchical FSMs automatically.

The relationships in use case charts go beyond those available in UML interaction overview diagrams. In particular, a scenario (or use case) may *preempt* or *suspend* another scenario (or use case). Scenarios (use cases) may have multiple concurrent executions. Scenarios (or use cases) may also be marked as failure cases, meaning that they stop execution. Finally, there is a well defined notion of negative scenario in EIODs.

The details of EIODs are not crucial to this paper and a full set of definitions can be found in [5]. For security modeling, however, we extend EIODs to include misuse cases and misuse scenarios. All the EIOD relationships (preemption, suspension, concurrent executions etc.) are also available for misuse modeling. We call such a diagram a misuse EIOD. Misuse EIODs introduce two new relationships—<<threatens>> and <<mitigates>> (following Alexander [1]). In a misuse EIOD, a misuse case is modeled as an activity node at level 1 and is stereotyped as <<misuse>>. A mitigation case is modeled as a level 1 node with the stereotype <<mitigation>>. A level 1 misuse can be refined at level 2 by a set of related misuse scenario nodes, each stereotyped with <<misuse>>. Similarly, the refinement of a mitigation at level 1 is by another set of <<mitigation>> nodes at level 2.

At level 3, a misuse scenario is formed of two parts. The first part, called the *modification scenario*, describes how an attacker modifies the system under development in order for his/her attack to succeed. The intuition is that sometimes an attacker may be able to change the core behavior of the system. For example, an attacker might tamper with a smartcard so that it ignores certain commands. A modification scenario is optional for a given attack. If it exists, the modification scenario will be modeled as an aspect sequence diagram. The second part of a misuse scenario is the *attack scenario*, which describes the malicious interactions of the attacker with the system. Attack scenarios are given as standard UML sequence diagrams except that attackers are stereotyped with <<attacker>>. There may be multiple attackers.

Level 2 mitigations are defined at level 3 by a *mitigation scenario*, described as an aspect sequence diagram.

The example in Section 2 had no modification scenarios but had one attack scenario (Figure 4(b)) and two mitigation scenarios (Figure 4(c)).

Levels 1 and 2 of a misuse EIOD may employ relationships from EIODs as well as <<threatens>> and <<mitigates>>. A <<threatens>> edge has a <<misuse>> node as its source and a use case node (or scenario node) as its target. A <<mitigates>> edge has a <<mitigation>> node as its source and a <<misuse>> node as its target.

The semantics of an EIOD is given by the set of event traces that it admits (see [5]). For misuse EIODs, the semantics is given by translating a misuse EIOD into the equivalent EIOD. This is done by weaving the aspect sequence diagrams (which can be modification scenarios or mitigation scenarios) with the core behavior scenarios. We describe this translation procedure for a level 2 misuse EIOD. The procedure can be extended to level 1 misuse EIODs in the natural way.

A level 2 misuse EIOD is a triple (N, E, s) where (N, E) is a directed graph of nodes and edges, and s is a function mapping each node onto the set of level 3 sequences defining the node. The nodes N are partitioned into use case nodes, misuse nodes and mitigation nodes, $N = U \cup M_s \cup M_t$. An edge is a triple (n_1, n_2, l) between two nodes, labeled by its edge type l. An edge (n_1, n_2, thr) where $n_1 \in M_s$, $n_2 \in U$ is a <<threatens>> edge. An edge (n_1, n_2, mit) where $n_1 \in M_t$, $n_2 \in M_s$ is a <<mitigates>> edge. For $n \in M_s$, $s(n)$ is partitioned into the set of modification scenarios $s_{mod}(n)$ and the set of attack scenarios $s_{att}(n)$.

For a misuse node, $n \in M_s$, we say that a level 3 sequence diagram, d, is in the scope of n if and only if there is a <<threatens>> edge from n to a node $m \in U$ and $d \in s(m)$. Similarly, for a mitigation node, $n \in M_t$, we say that a level 3 sequence diagram, d, is in the scope of n if and only if there is a <<mitigates>> edge from n to a misuse node, $m \in M_s$, and d is in the scope of m. Denote the scope of a node, n, by $\sigma(n)$.

To derive a level 2 EIOD from a level 2 misuse EIOD, firstly, for each misuse node, $n \in M_s$, weave each modification

scenario, $s \in s_{mod}(n)$, with each sequence diagram $d \in \sigma(n)$ and replace each d with $weave(d,s)$ in the misuse EIOD, where $weave(d,s)$ is the result of weaving s into d. Delete all modification scenarios. This step modifies the core behavior of the system according to the modification scenarios.

Secondly, for each mitigation node, $n \in M_t$, weave each mitigating scenario, $s \in s(n)$, into each sequence diagram, $d \in \sigma(n)$, in the scope of n and replace each d with the weaved sequence diagram, $weave(d,s)$. This step weaves the mitigating behaviors into the core behavior of the system. If all misuse and mitigation nodes are removed from the result, then we have an EIOD equivalent to the original misuse EIOD. The definition of weaving an aspect scenario into a sequence diagram is given in the next subsection. Note, however, that, in general, when there are multiple weaving steps, the order of weaving matters. Hence, the user must ultimately specify a weaving order. As described in [14], however, analysis techniques are available in MATA for determining ordering dependencies automatically.

Since modification and mitigation scenarios are given as aspect scenarios, they are crosscutting—in the sense that they cut across multiple core behavioral scenarios. Note that, according to the above translation procedure, the <<threatens>> and <<mitigates>> edges define the scope of this crosscutting. That is, they define the set of core scenarios crosscut by an aspect scenario. For example, to find the set of sequence diagrams crosscut by a mitigating scenario, one traverses the misuse EIOD from the mitigation node across the <<mitigates>> and <<threatens>> edges to a use case node, which is, in turn, defined by a set of sequence diagrams. These sequence diagrams are those that are crosscut by the mitigation scenario.

3.2 Modeling and Weaving Aspect Scenarios

Mitigations and modifications are modeled as aspect scenarios. This is done to promote reusability of security-specific concerns, which tend to crosscut core functional concerns. An aspect scenario is a scenario that crosscuts other scenarios. We model aspect scenarios using the aspect modeling language MATA [11]. In MATA, a *base* sequence diagram is crosscut by an *aspect* sequence diagram. The base is simply a UML sequence diagram. The aspect sequence diagram is written using a MATA profile for UML that includes the following stereotypes: <<create>>, <<delete>> and <<context>>.

In MATA, an aspect sequence diagram is a graph rule that defines a left-hand side (LHS) and a right-hand side (RHS). The LHS may contain variables and defines a pattern which captures the points in the base sequence diagram where the aspect will insert behavior (i.e., the join points). The RHS defines the behavior to be inserted and may refer to variables defined in the LHS. For sequence diagrams, new behavior in the RHS can be any modeling element used in a sequence diagram, including new messages, new lifelines, and new interaction fragments, etc.

The LHS and RHS are represented on the same diagram. If a model element in this diagram is stereotyped with <<create>> or <<delete>>, then it is part of the RHS. Otherwise, it is part of the LHS. The MATA tool weaves a base and aspect sequence diagram by finding a pattern match with the LHS and then inserting (or deleting) elements from the RHS for all matches. If an element on the RHS is marked as <<create>>, it is added to the matched base. <<delete>> removes an element from the matched base.

The <<context>> stereotype is a notational convenience used when there are container elements. For example, an **alt** fragment is a container because it contains interaction operands which, in turn, contain messages. To reduce the number of stereotypes that a modeler must write, if a <<create>> or <<delete>> stereotype is applied to a container element, then, by default, it is applied automatically to all model elements in the container. This default can be overridden by applying the <<context>> stereotype. Hence, if an **alt** fragment is marked with <<create>> but a message, m, inside one of its operands is marked with <<context>>, then m is not created, but instead is considered to be part of the LHS (i.e., it is matched against the base). An additional example is shown in Figure 6, which will be discussed shortly.

Figure 5 shows the result of weaving the mitigation aspect in Figure 4(c) with the base sequence diagram for voting in Figure

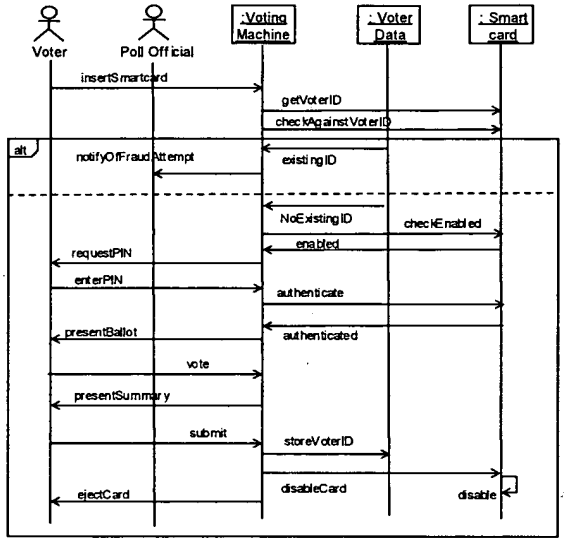


Figure 5: Composed Mitigation and Base Scenario.

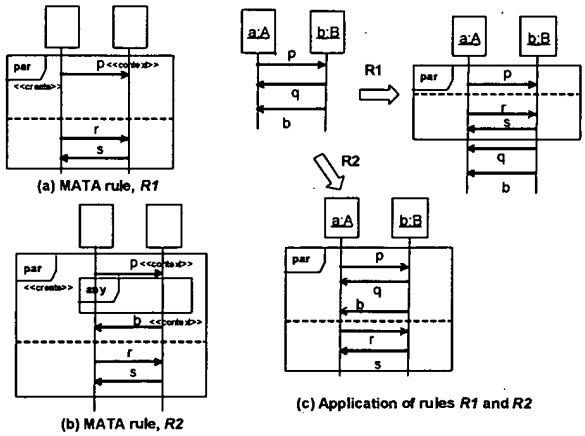


Figure 6: Sequence Pointcuts in MATA.

2(b). The effect is to insert the mitigation strategy into all places where it must apply.

Note that mitigation aspects require a much more expressive form of model weaving than is usually available in aspect-oriented modeling approaches. Typically, approaches allow behavior to be inserted *before*, *after* or *around* a base element, in a similar way to AspectJ. For sequence diagrams, this would correspond to adding a message before, after, or instead of a message in the base. This is not sufficient in our example because messages must be inserted at multiple places and an **alt** fragment must be added around existing base elements. In particular, note in Figure 4(c), that the second mitigation must insert an **alt** fragment so that an existing message (*submit)* is placed inside the second operand of the fragment. Also, a new *Voter Data* instance must be created. Weaving strategies such as these are impossible to specify simply by applying *before*, *after* and *around* advices to sequence diagrams.

MATA takes a unified approach to model weaving in that the same specification mechanism can be used for any modeling language, as long as it has a well-defined metamodel. In particular, weaving for UML class diagrams, UML sequence diagrams and UML state diagrams is done in the same way—the modeler uses pattern matching on the LHS of a graph rule to specify join points and specifies aspect behavior on the RHS using <<create>> and <<delete>>.

MATA aspect weaving is based on graph transformations and is thus easily automatable as well as amenable to formal analysis. Full details can be found in [11]. Briefly, the MATA tool takes a base model, given in UML, and translates it into a graph. Similarly, a MATA aspect is translated into a graph rule. The graph rule execution tool AGG [15] is used to execute these graph rules (i.e., weave the aspects). AGG is based on the theory of attributed typed graphs and a type graph of the source and target model defines the abstract syntax of graphs that can be transformed. This type graph is obtained in MATA by translating the UML metamodels into the type graph form recognized by AGG.

For modeling security concerns as aspects, we extended MATA to include *sequence pointcuts*. A sequence pointcut is a sequence of messages to be matched against in the base sequence diagram. In other words, the aspect will only apply if a collection of messages are found to occur in a particular order. Most AOM approaches for sequence diagrams allow only a single message to be a join point. It is therefore impossible to take into account the fact that messages may be causally related. This turns out to be an extremely useful feature for modeling security concerns, however, as can be seen by the matching of both *insertSmartcard* and *submit* in Figure 4(c).

By using sequence pointcuts, we can concisely capture all the messages in the base that are relevant to the aspect. Furthermore, we can abstract away from messages that must be matched against but that are not needed by the aspect, by using the new MATA **any** fragment. An **any** fragment will match against any number of messages. Since MATA is based on graph transformations, sequence pointcuts can be handled in a straightforward manner since they are just another application of pattern matching-based weaving. The tool implementation of MATA has been extended to include matching of **any** fragments using AGG as the back-end graph rule execution engine.

Figure 6 shows an example how sequence pointcuts can be defined in MATA. Rule R1 in Figure 6(a) is a simple aspect that does not involve a sequence pointcut. Since the two lifelines in the aspect have no names, they are MATA variables and can match any lifeline. Therefore, the aspect matches against a single message *p* between any two lifelines and creates a **par** fragment around *p* with two new messages, *r* and *s*, inserted into the second operand. (Note the use of <<context>> to override the <<create>> stereotype of the **par** fragment.) The result of applying aspect R1 to a simple base diagram can be seen in Figure 6(c).

Rule R2 in Figure 6(b) is a slight variation of rule R1 that uses a sequence pointcut to match against any *sequence* of messages that starts with *p* and ends with *b*. The messages occurring between *p* and *b* are unimportant to the aspect and so are matched against using the **any** fragment. (Note that since **any** fragments cannot be created by an aspect, they automatically are stereotyped with <<context>> if they occur inside a container stereotyped with <<create>> or <<delete>>.) The effect of applying R2 is different from R1 in that the messages *q* and *b* both appear inside the **par** fragment in the result—see Figure 6(c).

Sequence pointcuts turn out to be very convenient for modeling mitigation and modification scenarios as aspects. For the mitigation aspect in Figure 4(c), note how a sequence pointcut is used to match any sequence of messages coming after *insertSmartCard* and before *submit*. Sequence pointcuts allow all messages related to the mitigation aspect (in this case, storage of the voter ID) to be modeled on the same diagram.

3.3 Executing and Testing Misuse Cases

The previous subsection described how modification and mitigation scenarios are weaved into the core use case model to produce an EIOD that combines both the core behavior and the mitigation behavior. Since EIODs have a formal semantics, this resulting EIOD can now be executed, animated or analyzed. Currently, our tool supports animation but not formal analysis. However, off-the-shelf analyzers for FSMs could be used easily.

Recall that the attack scenario behavior is not included in the composed EIOD. This is because attack scenarios describe an attacker's behavior and therefore, its interactions do not form part of the system description. Instead, the attack scenarios are used as tests that can be automatically executed on the EIOD. If the tests fail, the EIOD is able to prevent the attacks. Otherwise, the mitigation strategies are not behaving as expected. Figure 7 outlines a recommended process for developing and testing executable misuse cases using the attack scenarios as a set of regression tests.

Executing the attack scenarios on the composed EIOD (step 3) is straightforward. Each attack scenario (i.e., UML sequence diagram) defines a set of event traces, so an attack scenario succeeds on an EIOD if its trace set is a subset of the traces described by the EIOD. Checking this, in practice, requires injecting events from the attack scenario into the EIOD animator.

Generally, events in the EIOD animator are uninterpreted. This can lead to nondeterminism when executing an EIOD, however. We saw an example of this in Section 2 where the animator needed to choose between the two events *existingID* and

NoExistingID. The modeler may provide pre/post-condition specifications for messages that define the semantics of the messages. The animator can then check which preconditions are enabled to choose between events, thus resolving any nondeterminism. In practice, we have found that it is enough to write pre/post-conditions over global Boolean valued variables. This is similar to the approach taken in [16]. Note that providing these specifications is optional. An alternative is to run the animator in interactive mode in which the animator queries the user as to which of the nondeterministic paths should be taken.

1. Model core system behavior as an EIOD
2. For each attack:
   a. Model attack as a modification scenario and attack scenario
   b. Compose modification scenarios into the core EIOD
3. Execute all previous attack scenarios on the composed EIOD.
   a. If an attack succeeds, debug and/or design a new mitigation strategy, compose into the core EIOD, and re-execute attack scenarios to check that the mitigations work as expected
   b. If all attacks fail, model next attack. Goto 2.

Figure 7: Procedure for Executable Misuse Cases.

4. EVALUATION

4.1 Methodology

In this section, we report on two case studies undertaken to evaluate the procedure of Figure 7. The models for the first case study, the EVS from Section 2, were developed by the authors but based on the attacks and system behavior described in [12]. Models for the second case study were developed by the third author. This second study concerns a positive train control system (PTC), an intelligent control system currently planned by the Federal Railroad Administration. The case studies validate only the suitability and executability of the modeling language. Further studies are necessary to evaluate the benefits of the technique in red teaming security requirements. The main result of this preliminary validation is that misuse EIODs are rich enough to express a wide range of realistic attacks and mitigations for real-world systems. In addition, the studies provide evidence that it is possible to animate precisely specified misuses in practice.

To quantify the range of attacks supported by our approach, we categorize attacks in each study according to two published classification schemes. A detailed taxonomy of attack types is presented in [17]. We limit ourselves to the portion of this taxonomy that catalogs attacks according to the *result* of the attack. In [17], there are five possible results: corruption of information, disclosure of information, denial of service, increased access and theft of resources. The second classification scheme is from the Information Assurance Technical Framework Forum (IATFF) [18] (sec. 1.3.5), which classifies attacks based on where the attacks occur. Attacks can be categorized as passive, active, close-in, insider, and distribution. *Passive* attacks come from external attackers who monitor and analyze system characteristics (e.g., network traffic) in order to obtain sensitive information. *Active* attacks come from external attackers who not only monitor a system but

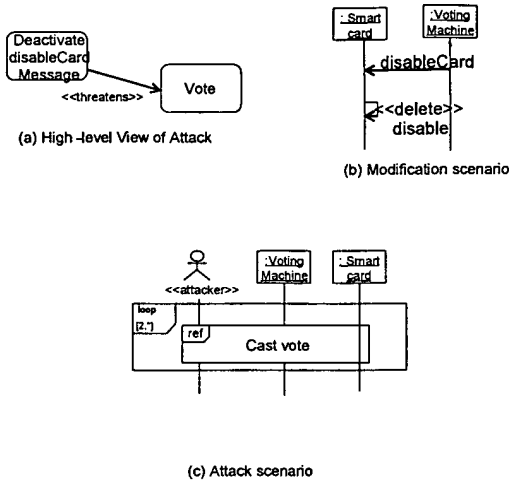


Figure 8: Attack 2–Deactivate Disable Card Message.

attempt to modify it by introducing malicious code or invoking denial of service. *Close-in* attacks are committed by attackers in close physical proximity to networks, systems or facilities. *Insider* attacks include action from both malicious insiders as well as non-malicious insider actions that result in undesired disclosures. Finally, distribution attacks focus on malicious modifications at the factory, for example, the inclusion of back doors into software.

4.2 EVS Results

We continue with the EVS example and demonstrate how to model an attack that includes a modification scenario. We then report on the overall results for the EVS system.

The second attack mentioned in [12] is a variation of the attack considered previously. Voting machines disable smartcards once a voter submits his/her vote. Therefore, if an attacker is able to reprogram a smartcard to ignore the disable message, s/he could reuse the card to cast multiple votes. Figure 8 shows the models for this attack. The misuse consists of both a modification scenario and an attack scenario. The modification scenario captures what an attacker must do to the EVS for the attack to succeed. S/he must modify the smartcard so that it ignores the *disableCard* message (see Figure 2). Figure 3(b) gives the FSM defining the behavior for smartcards. Therefore, for this modification, the FSM must be changed by the attacker. To model this, the modeler need only write the modification scenario, which will be weaved into the smartcard FSM to reflect the behavior of the smartcard once the malicious action of ignoring the *disableCard* message has been taken. The modifications are modeled using aspects so that they can be maintained separately from the rest of the model.

Figure 8(b) is the modification scenario. It looks for a pattern containing a *disableCard* directive sent from the voting machine to the smartcard. It then removes the action that disables the smartcard. This is done using MATA's <<delete>> stereotype which removes a model element from a base model. The attack scenario is the same as for attack 1 but is included in Figure 8 for completeness. Note again that it is of no concern to the model how the attacker manages to carry out this modification—such considerations are outside the scope of the model. Once the modification scenario is weaved into the core

behavior model, the weaved model captures the preparations that the attacker has made so that his/her attack can succeed. Note that there is a check in Figure 2(c) to see if the smartcard is enabled or not: a message *checkEnabled* is sent from the voting machine to the smartcard. For animation of the weaved FSM, the response of the smartcard should be linked with the *disableCard* action. As before, this can be done by adding a precondition of `enabled=true` to *checkEnabled* and a postcondition of `enabled=false` to *disableCard.*

No additional mitigation scenario is necessary for this particular attack since the mitigation for the first attack will also thwart this one. This is because the ID on the smartcard is unchanged when the attacker tries to reuse it, and so the check for an existing ID will still work. Note that the automation provided by executable misuse cases handles this automatically. In step 3 of the process in Figure 7, the modeler would execute the two attacks and find that both are mitigated. Hence, there is no need for a second mitigation scenario. Of course, with only two attacks, it is fairly straightforward to realize this without the benefit of automation. As the number of attacks increases, however, the benefits of automation become apparent.

Table 1 summarizes the attacks that were modeled for the EVS system. [12] discusses attacks arising both from design flaws and coding flaws. Only the design flaws are relevant for misuse modeling. In total, there were nine such attacks.

Table 1: EVS Attacks.

ID	Attack Name	Attack Category 1 [17]	Attack Category 2 [18]	Can be modeled?
1	Manufacture own smartcards	Corruption of Information	Active/ Close-in	Y
2	Deactivate disable card	Corruption of Information	Active/ Close-in	Y
3	Close voting prematurely	Denial of service	Active/ Close-in	Y
4	Modify ballot definition	Corruption of Information	Active	Y
5	Impersonate voting machine	Corruption of Information	Active/ Insider	Y
6	Tamper with election results	Corruption of Information	Active	Y
7	Link voters with votes	Disclosure of Information	Passive/ Insider	N
8	Tamper with audit logs	Corruption of Information	Active/ Insider	Y
9	Delay start of election	Denial of Service	Active	N

Table 1 shows a fairly wide range of attacks that can be modeled. As can be seen from the table, the attacks fall into three of the five categories for classification 1, and for classification 2, only distribution attacks are not covered.

Attack 7 could not be modeled because the attack only becomes apparent during low-level design since it concerns how data is stored. Attack 9 is a network-based attack. One would not expect a requirements modeling technique to be able to capture these attacks. Rather, these attacks would manifest themselves during detailed design, so it is desirable to continue the executable modeling of security concerns to later stages of the development lifecycle.

4.3 PTC Results

The second study modeled attacks for positive train control systems (PTCs), which are planned for deployment by the Federal Railroad Administration. PTCs [19] are distributed, interoperable wireless communications-based railroad control systems whose primary use cases aim to: (1) prevent train-to-train collisions, (2) enforce train speed restrictions, and (3) protect roadway workers. The functionality of a typical PTC was modeled by an EIOD that included five level 1 use cases. In a brainstorming session, a total of 19 attacks on these use cases were developed at a very high-level. It was realized that, of these 19 attacks, many were similar. Therefore, instead of modeling each attack individually, four representative attacks were chosen to be modeled. Models for the other 15 could easily be obtained, if desired, by modifying one of these four samples.

Table 2: PTC Attacks.

Attack ID	Attack Name	Attack Category 1 [17]	Attack Category 2 [18]	Can be modeled?
1	Modify Track Warrant	Corruption of Information	Active	Y
2	Jam Wayside Unit	Denial of service	Active	Y
3	Power Exhaustion Attack	Denial of service	Active	Y
4	Generic Insider attack	Corruption of Information/ Increased Access	Insider	Y

We briefly describe these four attacks here and summarize them in Table 2. PTCs rely on successful interactions between three classes of actors: the central dispatcher that directs railroad operations, mobile units (i.e., locomotives) that transport passengers and freight, and wayside units (e.g., signals). Though the individual subsystems in a PTC system are fail safe, the reliance of PTCs on wireless networks invites exploits that, unless explicitly addressed, may endanger rail operations. The first attack is a classic man-in-the-middle attack in which track warrants (authorizations issued from the dispatcher to mobile units) are intercepted by an attacker and modified en-route, thus potentially leading to collisions. In the misuse EIOD, this attack was modeled using a modification scenario that inserted messages into the core EIOD. These messages changed the data transmitted by a *sendMovementAuthority* message and also added messages so that mobile units would react adversely to the modified data. Additional authentication procedures were modeled aspect mitigation scenarios.

The second and third attacks were types of denial of service attacks—the second involved jamming wayside units and the third involved a power exhaustion attack on a wayside unit. Both of these could be modeled using a modification scenario to mimic the inability of wayside units to respond when under attack. A couple of mitigation scenarios were modeled. One involved the detection of the lack of responsiveness and subsequent rerouting of messages. The other introduced privileged frequencies for transmissions.

The fourth attack involved an insider at the dispatch office transmitting track warrants that caused collisions. The aspect mitigation scenario was to introduce a mediating authority. Table 2 classifies these four attacks.

4.4 Discussion

The two case studies are very different in nature. The PTC vulnerabilities arise mainly from the use of wireless communications. For the EVS, on the other hand, attacks commonly involve system access via modification of a physical resource (i.e., a smartcard).

Misuse EIODs tend to be most effective for modeling application-level security concerns. Network-level concerns can often be handled by standard encryption techniques and there seems to be limited benefit in modeling these precisely at the requirements level. Encryption could be handled in misuse EIODs, however, and it may be beneficial to maintain a library of misuses, such as network concerns, that can easily be adapted to a particular system. Misuse EIODs are well suited for modeling insider attacks. These attacks cannot be prevented by using standard encryption and require more complex mitigations such as monitoring and systems that periodically remove orphaned accounts. Misuse EIODs are also particularly well suited for modeling application-level or system-level attacks and could effectively be used, for example, to model more traditional, non-cyber security concerns, such as airport security.

Currently, misuse EIODs are a rather general modeling paradigm. This is both a strength and a weakness. It admits a wide range of attack types to be modeled. For security, however, there is a vast amount of domain knowledge available that could be brought to bear. For example, well known threat modeling methodologies (e.g., [20, 21]) advocate starting with generic versions of well known attacks, such as attack patterns. Incorporating attack patterns into misuse EIODs would improve the efficiency of misuse EIOD modeling. Right now, as with any modeling technique, a reasonable degree of expertise and experience is required to translate a high-level description of an attack into a precise misuse model. We believe, therefore, that a valuable addition to misuse EIODs would be to formalize well known attacks as generic patterns that could be instantiated to a specific application. In fact, it appears that aspects would be ideal for this purpose.

5. RELATED WORK

Misuse cases were first introduced by Sindre and Opdahl [2] and were developed further by Alexander [1]. Similarly, abuse cases [22] also consider a system from the viewpoint of what should be prevented. Misuse (and abuse) cases have received widespread attention because of their similarity to use cases and their intuitive nature. Despite this, support for misuse cases has been limited to defining the role of misuse cases in a software process [21], templates for describing misuse cases textually [23], and relatively simple tool support such as Alexander's ScenarioPlus [24]. These approaches succeed in promoting the utility of misuse cases but do so in a chiefly informal way. To the authors' knowledge, misuse EIODs are the first executable language for modeling with misuses.

The software industry has risen to the challenge of secure software engineering and there are a number of processes that attempt to address security concerns early in the development lifecycle. The most well known perhaps is Microsoft's threat modeling approach [20]. Others include the Security Quality Requirements (SQUARE) Methodology [25]. These are largely informal processes focusing on best practices and systematic guidelines. Misuse EIODs, however, are executable but could be used as a part of such processes.

Within requirements engineering, security has been considered by many existing requirements engineering methods. In [26], van Lamsweerde extends KAOS for modeling application-specific security requirements. Templates specify common security goals, where the goal is specified in first order logic using specialized predicates such as authorized, knows etc. Threats are modeled as (fault) trees, referred to as obstacle trees, in which the root is a negation of a security goal. The paper uses these anti-goals to derive objects and how attackers may use them in attacking the system under design.

Secure Tropos [27] is an agent-based design methodology for designing secure systems. It uses actors, roles, goals, tasks, resources, objectives, capabilities delegation and trust as basic design concepts. A goal represents strategic interests of the actor, and a task represents a course of action taken to satisfy a goal. The system uses permissions, delegating permissions based on trust etc. The main focus is on trust rather than adversary behavior.

Liu et al. [28] describe a social-science oriented requirements analysis method for secure systems. In this method, (a precursor to Secure Tropos), actors and attackers and their goals are identified. Then goals are refined to tasks, and their dependencies are specified. Mal-actor intent, and vulnerabilities are also identified, and their dependencies are used to derive the effectiveness of proposed counter measures. These concepts are formalized using Alloy.

Goal-based methods such as KAOS and Secure Tropos have not achieved as widespread use across industry as use cases. Therefore, we believe the two to be complementary. In fact, goals and scenario-based approaches can be effectively used together [29]. Goal models provide a systematic process for considering security goals and anti-goals at a high-level without showing detailed sequencing between goals. This sequencing can be provided by use and misuse cases. Also, since our misuse cases are executable, they permit automated red-teaming. Although some kinds of analysis can be done with goal models, the analysis often requires formal notations which are unlikely to be used by practitioners. It is an open question how best these two kinds of techniques can be used together. Note also that attack trees [30], which are used widely by industry, can be considered as a special case of goal models.

There has been a good deal of work on using aspects to represent security concerns. This has been chiefly at the design level, however, and so is complementary to the work in this paper. Kim uses a role-based metamodeling language to capture access control models [31] as aspects. Song et al [32] take a similar approach but consider security concerns as aspect sequence diagrams.

6. CONCLUSION

This paper presented a new technique for precisely modeling and executing misuse case scenarios for secure systems development. The work integrates previous research on executable modeling with scenarios and on weaving aspect scenarios. However, the paper makes contributions to both of these areas as well as applying the integration of the two techniques to a new application area. The result is a modeling approach that allows stakeholders to brainstorm potential security attacks, capture mitigations of those attacks, and test whether the attacks have indeed been thwarted.

Future work will investigate how to transform the results of analysis into later design stages. We will examine how to apply a model-driven approach to develop secure designs in such a way that the separation of core behavior and mitigation behavior, which was achieved at the requirements level, can be maintained during design. We also plan to understand better the exact relationship between goal-based models, such as attack trees, and misuse case models. Finally, we will investigate how to support misuse case modeling, for example, by generating attack sequences automatically.

7. ACKNOWLEDGMENTS

Thanks to Amr Said who implemented MUCSIM.

8. REFERENCES

[1] I. Alexander, "Misuse cases: use cases with hostile intent," *IEEE Software*, vol. 20, pp. 58-66, Jan./Feb. 2003.
[2] G. Sindre and A. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, pp. 34-44, Jan. 2005.
[3] I. Alexander, "Initial Industrial Experience of Misuse Cases in Trade-off Analysis," in *International Conference on Requirements Engineering* Essen, Germany: IEEE Computer Society, pp. 61-70.
[4] J. Whittle, "Specifying Precise Use Cases with Use Case Charts," in *MoDELS Satellite Events (Revised Selected Papers)*. vol. 3844, J.-M. Bruel, Ed. Montego Bay, Jamaica: Springer-Verlag, pp. 290-301.
[5] J. Whittle, "Precise Specification of Use Case Scenarios," in *Fundamental Approaches to Software Engineering (FASE07)*. vol. LNCS 4422 Braga, Portugal: Springer, 2007, pp. 170-184.
[6] J. Araújo, J. Whittle, and D.-K. Kim, "Modeling and Composing Scenario-Based Requirements with Aspects," in *International Conference on Requirements Engineering* Kyoto, Japan, 2004, pp. 58-67.
[7] J. Whittle, A. Moreira, J. Araújo, R. Rabbi, P. Jayaraman, and A. Elkhodary, "An Expressive Aspect Composition Language for UML State Diagrams," in *International Conference on Model Driven Engineering, Languages and Systems (MODELS)*, Nashville, TN, 2007, pp. 514-528.
[8] J. Whittle, J. Araújo, and A. Moreira, "Composing Aspect Models with Graph Transformations," in *Workshop on Early Aspects at ICSE*: ACM Press, 2006, pp. 59-65.
[9] J. Whittle and P. Jayaraman, "Generating Hierarchical Finite State Machines from Use Case Charts," in *14th IEEE International Conference on Requirements Engineering (RE'06)* Minneapolis, 2006, pp. 16-25.
[10] P. Jayaraman and J. Whittle, "UCSIM: A tool for simulating use case scenarios," in *Companion to the 29th International Conference on Software Engineering (ICSE)* Minneapolis, 2007, pp. 43-44.
[11] J. Whittle and P. Jayaraman, "MATA: A Tool for Aspect-Oriented Modeling based on Graph Transformation," in *Workshop on Aspect Oriented Modeling at the International MODELS Conference*, Nashville, TN, 2007.
[12] T. Kohno, A. Stubblefield, A. Rubin, and D. Wallach, "Analysis of an Electronic Voting System," in *IEEE Symposium on Security and Privacy*: IEEE Computer Society Press, 2004, pp. 27-40.
[13] OMG, "Unified Modeling Language 2.1.1 Specification (05-07-04)," http://www.omg.org, Specification 2007.
[14] P. Jayaraman, J. Whittle, A. Elkhodary, and H. Gomaa, "Model Composition in Product Lines and Feature Interaction Detection using Critical Pair Analysis," in *International Conference on Model Driven Engineering, Languages and Systems (MODELS)*, Nashville, TN, 2007.
[15] G. Taentzer, "AGG: A Graph Transformation Environment for Modeling and Validation of Software," in *Conference on Applications of Graph Transformations with Industrial Relevance (AGTIVE)*, Charlottesville, VA, 2003, pp. 446-453.
[16] J. Whittle and J. Schumann, "Generating statechart designs from scenarios," in *22nd International Conference on Software Engineering* Limerick, Ireland: ACM Press, 2000, pp. 314-323.
[17] J. Howard and T. Longstaff, "A Common Language for Computer Security Incidents," Sandia National Laboratories 1998.
[18] "Information Assurance Technical Framework (IATF), Release 3.1," Information Assurance Solutions, US National Security Agency, Fort Meade, MD Sep. 2002.
[19] M. Hartong, R. Goel, and D. Wijesekera, "Use Misuse Case Driven Forensic Analysis of Positive Train Control: A Preliminary Study," in *2nd IFIP WG 11.9 International Conference on Digital Forensics* Orlando, FL.
[20] F. Swiderski and W. Snyder, *Threat Modeling*: Microsoft Professional, 2004.
[21] G. McGraw, *Software Security: Building Security In*: Addison Wesley, 2006.
[22] J. McDermott, "Using abuse case models for security requirements analysis," in *15th Annual Computer Security Applications Conference*, 1999, pp. 55-64.
[23] G. Sindre and A. Opdahl, "Templates for Misuse Case Description," in *7th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ)* Switzerland, 2001.
[24] I. Alexander, "Scenario Plus Use Case Toolkit (http://www.scenarioplus.org.uk/)," 2005.
[25] N. Mead, "Identifying Security Requirements Using the SQUARE Method," *Integrating Security and Software Engineering: Advances and Future Visions*, pp. 44-69, 2006.
[26] A. van Lamsweerde, "Elaborating Security by Construction of Intentional Anti-Models," in *International Conference on Software Engineering (ICSE)*, 2004, pp. 148-157.
[27] F. Massaci, J. Mylopoulos, and N. Zannone, "Computer-Aided Support for Secure Tropos," *Automated Software Engineering*, vol. 14, pp. 341-364, 2007.
[28] L. Liu, E. Yu, and J. Mylopoulos, "Security and Privacy Requirements Analysis within a Social Setting," in *11th IEEE International Requirements Engineering Conference* Monterey Bay, CA, 2003.
[29] L. Liu and E. Yu, "From Requirements to Architectural Design: Using Goals and Scenarios," in *ICSE 2001 Workshop: From Software Requirements to Architectures*, pp. 22-30.
[30] B. Schneier, "Modeling Security Threats," in *Dr. Dobb's Journal*, 1999.
[31] D. K. Kim, "A Pattern-Based Technique for Developing UML Models of Access Control Systems," in *30th Annual International Computer Software and Applications Conference (COMPSAC)* Chicago, IL, 2006, pp. 317-324.
[32] E. Song, R. Reddy, R. B. France, I. Ray, G. Georg, and R. Alexander, "Verifiable Composition of Access Control and Application Features," in *ACM Symposium on Access Control Models and Technologies (SACMAT)* Stockholm, Sweden, 2005, pp. 120-129.

Providing Granted Rights with Anonymous Certificates

Michael Maaser, Steffen Ortmann
IHP microelectronics
Frankfurt/Oder, Germany
{maaser|ortmann}@ihp-microelectronics.com

Abstract— **The electronic media of today requires users to authenticate or provide identifying properties which is usually done with login credentials or to present digitally signed certificates. Certificates are issued by a trusted certificate authority, which knows the content of a certificate and the identity of the owner. To provide a maximum level of privacy the identity of the owner must be separated from the certificate. Neither the service shall be able to determine the identity of the owner nor shall the certificate authority track the certificate throughout its use. The certificate authority may issue certificates with a known content for a pseudonymous identity that cannot be linked to the real identity of the owner. Hence the content is to be openly signed whereas the pseudonym is signed blindly. Both items have to be interweaved to ensure that the content is valid for this pseudonym only as well as to prevent from forging a certificate. After unblinding the pseudonym the certificate can be presented for use. Now the service can validate the content and the user can prove his ownership of the pseudonym in the certificate. Even with collaboration between the service and the certificate authority it is impossible to map the presented certificate to a certain identity. Thus, our approach allows a fully anonymous use of services.**

I. Introduction

Usually access control goes along with identification and authentication. This process often requires to provide proof of identity or to verify released personal data. In real life proof can be provided with confidential documents like passports, whereas electronic media prefers the use of credentials or digital certificates for authentication. Digital certificates are trustworthy documents that are issued by a trusted certificate authority (CA) where the user has been identified through official documents. Certificates issued by a CA can be used as digital identity cards for electronic services. Given the user data and his certificate a service can request the CA to check the user data. For example an age-restricted pay-TV service could verify the age of a user by his certificate and thereby authorize him to use the service. Such a verification process not only releases the age of the certificate's owner to the service but all of the other information contained in the certificate as well. Furthermore, the CA acquires knowledge about the usage of the certificate thus making an anonymous usage of certificates infeasible. In consideration of privacy, a service should only receive the essential information and the CA does not need to know where, when and for what an issued certificate is used for. Our certification method provides granted rights by using fully anonymous individual-related certificates. The content and the correctness of any issued certificate are preserved while the legal ownership can be proved anonymously. In the pay-TV example, neither the CA can find out who consumed which content nor is the service able to map certificates to real identities, not even in mutual collaboration.

The rest of this paper is structured as follows. We start with a short overview on related research. Section III introduces the function principle of our approach and presents the basic math. Finally our paper concludes with a summary and an outlook on further work.

II. Related Work

There exist many approaches that consider privacy and anonymity for trustable electronic media. The major goal is to establish authentic user data and authentication methods for service usage. Chaum developed the concepts of credential system [1] and group signature schemes [2] to establish verifiability on user data. Credential systems are made to give a proof of identity when interacting with electronic services. Signatures are used to prevent electronic documents from being changed by unauthorized entities. Nowadays, certificates are the preferred means for authorization and identification issues. Due to certificates' increased use it is important to provide better privacy and anonymity for users.

Most approaches focus on enhancing the user's privacy towards services, e.g. by using pseudonyms in certificates and e-cash schemes as provided in [3], [4]. Releasing data implies a trusted third party, for example a CA or a bank, where the pseudonym can be mapped to the real identity of the user. Services may request the trusted third party to verify the correctness of certified data for a given pseudonym. Such approaches support anonymity of users against services but enable the trusted third parties to track issued certificates. Even privacy enhancing certification frameworks as presented in [5] and [6] still use a trusted third party and do not provide a solution for fully anonymous granting of certificates. Preventing trusted third parties from tracking issued certificates either needs to use untraceable pseudonyms or blind the content of the certificate before signing. Chaum published the idea of blinding certificates [7] that allows concealing the content from the CA. Therein the CA is requested to issue a certificate with unknown content. Consequently this method is inefficient particularly with regard to verifiability of certified data. The

978-1-4244-2182-4/08/$25.00 ©2008 IEEE.

CA might issue invalid certificates or could grant certificates for unauthorized identities.

Even a combination of known approaches would not bear up against a collaborated attack of services and trusted third parties. To the best of our knowledge there is no known approach, that provides fully anonymous use of certificates where neither the service nor the CA is able to map the certificate to a real identity. Nevertheless our approach can guarantee that only valid certificates are issued by the CA.

III. Function Principle

This chapter sets up the means for a privacy aware certification system. In such system a user retrieves pseudonymous certificates for rights legally granted to him. Thereby the CA does not get hold of the pseudonym. Services gain the pseudonym from verification of the presented certificates by the CA. Then the user proves ownership of the pseudonym to the service.

A. Requirements

Supposed that a user has certain rights, either by explicit grant or due to physical properties, e.g. age. If a service requires certain rights those must be presented by a potential user. Usually this is done by authenticating the user whereby the service knows which rights are granted to him. We do not require a user to be known to the service he uses. Thus, the user has to present the granted rights explicitly. To ensure that a user presents valid rights only, those rights must be issued by a trusted authority, which knows all users and their respective granted rights. Rights shall be revocable or valid for a limited time. It is not required to prevent the use of a certified right multiple times within its valid time. To prevent eavesdropping and the unallowed use of certificates by other users they are issued to a specific user. Even though a certificate is issued to a specific user it must not be possible to track it. For privacy reasons the CA shall not be able to know which user presented a particular right certificate to a service. Hence, a right certificate must contain:

- a verifiable right
- a verifiable expiration date
- a means to prove ownership, which cannot be mapped to the user identity by the service or the authority

As proof of ownership a pseudonym in form of a public-private key pair is used. The certificate contains the public key whereas the user possesses the private key of that pair. Since the certificate shall not be mapped to the user's identity the authority must not know the pseudonym when issuing the certificate. The way to certify something without knowing it is blind signing [7]. The downside of blind signing is that the signer has no means to verify what he actually signs. It is not reasonable to sign a right or an expiration date in blind. Otherwise the authority might sign a right that is not granted or for an unrealistic expiration date, which breaks the possibility of right revocation. A means to interweave the openly signed right and the blindly signed pseudonym is shown in the following section.

B. Math

The following shows the math of the algorithm. We chose to number the rights and the possible expiration dates, e.g. days from 2007-01-01. The blinded pseudonym is signed r times with a key dedicated for rights and t times with a different key for expiration dates, being r the number of the right to be certified and t the expiration date of the certificate. We are aware of the fact that a malicious user could try to get a pseudonym signed multiple times to gain rights with higher numbers that are not actually granted. Prevention of this illegal right acquisition is presented in Section C. Encrypting a certain message r times and afterwards decrypting it r times with the respective keys of one RSA key pair will result in the same original message (see (1) to (4)). Therein the public key is $(e; N)$, the private key is $(d; N)$ and the message to be encrypted is m.

$$ed \equiv 1 \; mod \; \varphi(N) \tag{1}$$

$$\Rightarrow \quad m^{ed} \equiv m \; mod \; N \tag{2}$$

$$e^r d^r \equiv 1^r \; mod \; \varphi(N) \tag{3}$$

$$\Rightarrow \quad (m^{e^r})^{d^r} \equiv m^{(ed)^r} \equiv m \; mod \; N \tag{4}$$

Let the user randomly choose a pseudonym p and blind factor k secretly. With k and the publicly known modulus of the signature key of the CA he can calculate the modulus inverse k^{-1} with

$$kk^{-1} \equiv 1 \; mod \; N \tag{5}$$

According to Chaum, the pseudonym or a hash value of it can be blinded by modular multiplication with the encrypted blind factor. For encrypting k the public key of the CA is used. This product is given to the CA to be signed with the respective private key. Without actually knowing the factors in the product this results in an encryption of the pseudonym and a decryption of the blind factor which can be neutralized by multiplication with the inverse k^{-1}. In (6) we can find the result of an openly signed pseudonym and compare it with the result of the blind signing (9). Formulas (7) and (9) are processed at the user's side and (8) at the authority's side. It can easily be seen, that the authority does not get to know the actual pseudonym p nor the signed pseudonym s. At the same time the user gets a signed pseudonym, without knowing the authority's private key $(d; N)$.

$$p^d \equiv s \; mod \; N \tag{6}$$

$$p(k^e) \equiv p_{blind} \; mod \; N \tag{7}$$

$${p_{blind}}^d \equiv p^d(k^{ed}) \equiv p^d k \equiv s_{blind} \; mod \; N \tag{8}$$

$${s_{blind}}k^{-1} \equiv p^d k k^{-1} \equiv p^d \equiv s \; mod \; N \tag{9}$$

In Chaum's blind signature the information is just: "Yes, this has been signed with the signer's private key." Our intention is to let it say: "This has been signed with the signer's private keys for right r or other stated purpose." Therefore we extended the blind signature procedure in two ways. The first is to sign the blinded pseudonym multiple times. The count of signings encodes separate information, i.e. the right. This

way the authority gains some control of what it signs. Still there is no knowledge about the pseudonym that is signed. Only, decrypting it exactly the number of times that it was signed results in the pseudonym, which can then be verified. If it cannot be verified it is either not the pseudonym of the presenting user or was not decrypted the right number of times and hence not certifying the right it claims to.

$$p(k^{e^r}) \equiv p_{blind} \; mod \; N \quad (10)$$

$$p_{blind}{}^{d^r} \equiv p^{d^r} k^{(ed)^r} \equiv p^{d^r} k \equiv s_{blind} \; mod \; N \quad (11)$$

$$s_{blind} k^{-1} \equiv p^{d^r} k k^{-1} \equiv p^{d^r} \equiv s \; mod \; N \quad (12)$$

The operations in (11) are processed in the CA with knowledge of neither the pseudonym nor the blind factor k. The operations in lines (10) and (12) are executed on the user's side which generates a signed pseudonym without knowing the private key of the authority.

The second extension of the procedure is to use multiple key pairs to encode orthogonal information, i.e. the right and the expiration date. There is only one constraint on the keys to be used. All have to use the same modulus. Formulas (13) to (15) show the process of creating a blind signature with two key pairs $[(e;N),(d;N)$ and $(\hat{e};N),(\hat{d};N)]$. This can easily be extended to more than two key pairs as long as they use the same modulus.

$$p(k^{(e^r \hat{e}^t)}) \equiv p_{blind} \; mod \; N \quad (13)$$

$$p_{blind}{}^{d^r \hat{d}^t} \equiv p^{d^r \hat{d}^t} k^{(ed)^r (\hat{e}\hat{d})^t} \equiv p^{d^r \hat{d}^t} k \equiv s_{blind} \; mod \; N \quad (14)$$

$$s_{blind} k^{-1} \equiv p^{d^r \hat{d}^t} k k^{-1} \equiv p^{d^r \hat{d}^t} \equiv s \; mod \; N \quad (15)$$

Using both extensions of the blinding algorithm the user can get a signed pseudonym as if it was signed using the private keys of the certificate authority but without letting the authority know the pseudonym. On the other hand the authority can ensure that the signature is used for the intended purpose only, because the signature turns into a valid pseudonym only if decrypted with the right numbers of times r and t of the public keys $(e;N)$ and $(\hat{e};N)$.

There is a chance that the pseudonym could be decrypted correctly even when applying a wrong right or expiration date. This is called a collision. A collision may occur when

$$e^{r_1} \hat{e}^{t_1} \equiv e^{r_2} \hat{e}^{t_2} \; mod \; \varphi(N). \quad (16)$$

With e and $\hat{e}$ being relatively prime to $\varphi(N)$ they and their powers belong to the multiplicative group of integers modulo $\varphi(N)$. The likelihood for finding two congruent pairs of $e^{r_1}\hat{e}^{t_1}$ and $e^{r_2}\hat{e}^{t_2}$ is the reciprocal to the order of that group, which is $\varphi(\varphi(N))$. According to [8] $\varphi(N) \geq \sqrt{N} \; \forall N > 6$. Hence, the likelihood of colliding rights and expiration dates is equal or less than $\frac{1}{\sqrt[4]{N}}$. For bit lengths of N at 1024 or more this becomes negligible.

C. Malicious Attacks

Up to here the basic approach is still vulnerable to attacks like fraudulent falsification and eavesdropping. So by requesting blind signature for certain granted rights multiple times he is able to gain a right, which was not actually granted to him. That is, a user might be granted the rights r_1 and r_2 whereas a right $r_1 + r_2$ was not granted. Since the CA has no means to check the content of the pseudonym to be signed it does not notice when signing an already signed pseudonym see (17) to (19).

$$p^{d^{r_1}} k^{e^{r_2}} \equiv p_{blind} \; mod \; N \quad (17)$$

$$p_{blind}{}^{d^{r_2}} \equiv p^{d^{r_1+r_2}} k^{(ed)^{r_2}} \equiv p^{d^{r_1+r_2}} k \equiv s_{blind} \; mod \; N \quad (18)$$

$$s_{blind} k^{-1} \equiv p^{d^{r_1+r_2}} k k^{-1} \equiv p^{d^{r_1+r_2}} \equiv s \; mod \; N \quad (19)$$

A similar attack is gaining a not granted right with a lower ID number than a given granted right. Assume a right r_1 is granted whereas the right $r_2 = r_1 - 1$ is not granted. Once the user possesses a pseudonym signed for r_1 he can create a valid one for r_2 by decrypting it once with the public key of the CA, see (20).

$$p^{d^{r_1} e} \equiv p^{d^{r_1 - 1}} \; mod \; N \quad (20)$$

To tackle these issues, both keys of a pair are used privately to the CA and only the modulus is shared with users to prevent from reduction of the certified right. The private-private key pair $(\bar{e};N),(\bar{d};N)$ is used to flag the number of signature steps. Independent of the right to be certified the blinded pseudonym gets signed exactly once with $(\bar{d};N)$. In case the attacking user tries to get a signature for an already signed pseudonym, this key $(\bar{d};N)$ is applied more than once. Hence, an attack is detected while verifying the pseudonym's rights.

Since here all keys are private the blind factor has to be encrypted by the CA (21). On the other hand the CA must not know the applied blind factor or it could determine the pseudonym. It is feasible to use the CA-encrypted blind factor to the power of n modulo N with n being randomly chosen and unknown to the CA (22). After the blind signing (23) only k^n remains for which the modular inverse k^{-n} can easily be determined by the user (24).

$$k^{\bar{e}e^r} \equiv k_{CA-enc} \; mod \; N \quad (21)$$

$$p(k_{CA-enc}{}^n) \equiv p_{blind} \; mod \; N \quad (22)$$

$$p_{blind}{}^{\bar{d}d^r} \equiv p^{\bar{d}d^r} k^{\bar{e}\bar{d}(ed)^r n} \equiv p^{\bar{d}d^r} k^n \equiv s_{blind} \; mod \; N \quad (23)$$

$$s_{blind} k^{-n} \equiv p^{\bar{d}d^r} k^n k^{-n} \equiv p^{\bar{d}d^r} \equiv s \; mod \; N \quad (24)$$

In fact CA-encrypting the blind factor and signing the blinded pseudonym are the corresponding en- and decryption operations. This could be exploited to remove one flag, which reflects the number of signature steps, when signing a pseudonym twice. To prevent this we use a pre-shared generator as k. Since now the service does not know the authority's keys the decryption and verification has to be forwarded to the CA. If a user presents the signed pseudonym to a service claiming that it was issued for right r and expiration date t the CA decrypts the signed pseudonym r times with $(e;N)$, t times with $(\hat{e};N)$ and once with $(\bar{e};N)$ as shown in (26).

$$s \equiv p^{\bar{d}d^r \hat{d}^t} \; mod \; N \quad (25)$$

$$s^{\bar{e}e^r \hat{e}^t} \equiv p^{\bar{d}d^r \hat{d}^t \bar{e}e^r \hat{e}^t} \equiv p \; mod \; N \quad (26)$$

The service has no information on the identity of the user and hence the CA does not get to know the identity either. Since the CA has never known the signed pseudonym, which is un-blinded during the signature process, it cannot map it to an identity in the verification process. The CA returns the pseudonym to the service if and only if the decrypted result for the claimed right, expiration time and exactly one flag is in fact a valid pseudonym. This can be achieved by checking the format of the pseudonym. We use the public key of an arbitrary key pair $((\tilde{e};U),(\tilde{d};U))$ concatenated with a secure hash of itself as pseudonym. This enables the CA to verify the format of the decrypted pseudonym. If the format cannot be verified the certificate has probably been claimed with incorrect parameters (fraudulent use of certificate). Further the public key $(\tilde{e};U)$ can be used to encrypt a challenge (27) that only the owner of the private key $(\tilde{d};U)$ belonging to the pseudonym can decrypt (28). If the user cannot decrypt the challenge, he is not the owner of the pseudonym (eavesdropped certificate).

$$c_{hallenge}{}^{\tilde{e}} \equiv c_{enc} \bmod U \tag{27}$$

$$c_{enc}{}^{\tilde{d}} \equiv c_{dec} \equiv c_{hallenge} \bmod U \tag{28}$$

D. Implementation Issues

According to the algorithm described, the exponents in the used RSA keys have to be raised to the power of r or t. With this not being a modular operation this power increases very quickly. Assuming an exponent in an RSA key of about 512 bit and the right 13 the resulting power to be used in the modular exponentiation has a length of 6656 bit. For expiration dates this length can be even much higher. On one hand these huge bit-lengths of exponents consume unfeasibly amounts of memory and computation time in their calculation. Note, raising the encryption exponent to a power of n means exactly the same as encrypting n times. Thus, we could sequentially execute n modular operations using only the encryption exponent.

$$m^{(e^3)} \equiv ((m^e)^e)^e \bmod N \tag{29}$$

The sequential execution of n modular operations definitely decreases the amount of used memory but rather increases the calculation effort. In our approach all encryption operations, which use an encryption exponent raised to a power, are executed in the CA, which also knows $\varphi(N)$. Using the dependency in (30) the calculation of the encryption exponent can be reduced from a memory and time consuming calculation of e^n to $e^{n \bmod \varphi(N)}$. Using the reduced exponent for encryption results in the same value as if using the very large power or encrypting n times using the original exponent e (see (31)).

$$x \equiv y \bmod \varphi(N) \Rightarrow m^x \equiv m^y \bmod N \tag{30}$$

$$m^{(e^n)} \equiv m^{(e^n \bmod \varphi(N))} \bmod N \tag{31}$$

The calculation effort for this reduction is the same as one regular encryption with one key. So the effective calculation effort for encrypting n times using the original exponent reduces from n to 2 operations. This reduction has to be accomplished for each key in use $e, \hat{e}, \bar{e}$. Hence, for the authority the encryption effort does not grow with the numbers r or t to be encoded, but linearly with the number of different keys to be used.

IV. Conclusion & Outlook

A means to provide granted rights and properties without revealing the identity of the accessing entity was presented. Granted rights or properties are presented to services or access controls as digitally signed certificates, which cannot be forged. The certificates are signed by a trusted authority that knows the identity and hence the rights and properties of the user, but is not able to track the issued certificates. The certificates are signed in blind, which protects the user's privacy from the authority. Thereby it can be ensured that some parts, e.g. the right and the expiration date, of the certificates are verifiable by the authority on behalf of the services or other entities. The verifiable information is inseparably woven into the blindly signed certificate to prevent a combination of false claims and valid signatures. The ownership of a certificate can be proven by the presenter without releasing his identity. This protects the privacy of the user against the service he uses. The math in our approach has been proven. Within a current project we implemented the certification algorithm using the extended blind signature. The implementation discovered performance issues, which we presented together with a respective solution. In future research we intend to transfer our approach to elliptic curve cryptography. With elliptic curves less bit length is needed for the same level of security. We also expect to reduce the calculation effort on the users side, especially on mobile devices with limited capability. Offline verification by the service might even be enabled.

References

[1] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," in *Communications of the ACM 24*. ACM, February 1981, pp. 84–88.

[2] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology*, EUROCRYPT '91. Springer-Verlag, 1991, pp. 257–265, vol. 547 of LNCS.

[3] S. Brands, "Rethinking public key infrastructure and digital certificates - building in privacy," Ph.D. dissertation, Institute of Technology, Eindhoven, Netherland, 1999.

[4] K. Piotrowski, P. Langendörfer, and D. Kulikowski, "Moneta: An anonymity providing lightweight payment system for mobile devices," in *Proceedings of the 2nd International Workshop for Technology, Economy, Social and Legal Aspects of Virtual Goods*, 2004.

[5] E. Bangerter, J. Camenisch, and A. Lysyanskaya, "A cryptographic framework for the controlled release of certified data," in *Twelfth International Workshop on Security Protocols 2004*. Springer-Verlag, 2004, lNCS.

[6] J. Camenisch, D. Sommer, and R. Zimmermann, "A general certification framework with application to privacy-enhancing certificate infrastructures," in *Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006)*, Karlstad, Sweden, May 22-24 2006.

[7] D. Chaum, "Blind signature systems," in *Advances in Cryptology*. CRYPTO '83, 1984.

[8] D. G. Kendall and R. Osborn, "Two simple lower bounds for euler's function." *Texas Journal of Science*, vol. 17, no. 3, 1965.

Networking in The Solar Trust Model: Determining Optimal Trust Paths in a Decentralized Trust Network

Michael Clifford
The Aerospace Corporation
2350 East El Segundo Blvd.
El Segundo, CA 90245
clifford@aero.org

Abstract

The Solar Trust Model provides a method by which the sender of a message can be authenticated, and the level of trust that can be placed in the sender of the message or the message itself can be computed. The model works even if there is no prior relationship between the sender and receiver of the message. The Solar Trust Model overcomes a variety of limitations inherent in the design of other trust models and public key infrastructures. This paper presents a variety of enhancements and formalizations to the basic concepts of the model. In addition, this paper provides a set of algorithms that can be used to determine all of the possible trusted paths along which a message can be sent from a sender to recipient and the optimal choice of paths from a selection of paths. The paper also presents algorithms for reducing the network load produced by the model through piggybacking, path caching, and load distribution techniques.

1. Introduction

Electronic commerce requires both authentication and trust. For example, in order to make a purchase over the Web, authentication is required so that the purchaser of goods or services can be sure of with whom they are talking. Trust is required in order for the purchaser to feel confident that they will actually receive the goods that they purchased, and for the supplier to be able to believe that they will actually receive their money in exchange for their product. Likewise, in many other fields, it is critical that the recipient of a piece of information be able to tell both who generated that information and whether or not the information is likely to be accurate. For example, a patient who obtains medical advice must be confident not only that they are talking to a doctor, but that the doctor also has sufficient skill and experience to be able to diagnose their problem correctly. Traditional trust and PKI models offer either limited trust computation or authentication services, but not both.

Traditional models are also limited by a uniform notion of trust that assumes that all users of the model will regard trust in exactly the same way or will trust a single ultimate Trust Authority. Because user requirements for trust can vary significantly depending on the user and context, a truly functional trust model must be able to determine a level of trust that meets each user's unique requirements, and to adapt easily to changing conditions. It is not realistic to assign an exact numerical value to a level of trust or to use a universal formula to make such a computation, since this would force the user to accept either a judgment about a level of trust that they may not agree with, or a definition of trust that is imposed from outside. Rather, it only makes sense for the end user to determine an ordering of trust, and to make judgments about a particular situation relative to that ordering. This notion of trust is supported by [14].

The PGP/X.509 model [1] assumes an implicit transitivity of trust between participating parties. The PEM certification model [2] assumes that everyone in the world trusts one ultimate authority to verify the identities of other certificate senders. The PEM model also does not allow for multiple levels of trust within its certification hierarchy. Neither of these models provides a practical, universally applicable method of verifying the identity of the sender of a message. In the real world, trust is rarely transitive, and the concept of a single hierarchy has already given way to multiple corporate and governmental certifying authorities, each of which issues their certificates without being required to meet the standards of some ultimate certifying authority (CA) [3]. The Biba Integrity Model [4] provides for multiple levels of trust within an organization or

Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC'02)
1063-9527/02 $17.00 © 2002 IEEE

system, but not between two autonomous systems. The Common Security Services Manager's Trust Policy Interface Specification [5] provides an API in which an open set of authentication policies may be implemented, and permits multiple certificate authorities to sign the same message, but provides no method for verifying the trustworthiness of any of the CAs. The ICE-TEL trust model [6] eliminates the need for transitivity of trust, and permits certification along paths rather than in global CA hierarchies, but it does not provide for multiple levels of trust. The limitations on these models are discussed in great detail in [7], which introduced the concept of the Solar Trust Model.

The Solar Trust Model [7] enhances verification capabilities beyond other models by providing a method for designating many levels of trust, for permitting unlimited numbers of independent authenticators with no requirement for a central authority, and for determining the trustworthiness of a message that has been signed by an authenticator with whom the receiver of the message has no direct relationship. Each end user is able to judge on the fly a level of trust that can be assigned to an interaction, event, or other entity. These judgments are made relative to other previously made judgments, and are always made in a way which is appropriate to the context of the situation.

This paper expands on the work in [8,9,10] by describing new algorithms for the discovery of trusted paths, choosing optimal paths, and for reducing network load through caching and piggybacking methods. Additionally, a number of refinements and enhancements to the Solar Trust Model are presented, and the entire model is described in a more formal manner than has been done previously.

2. Background

The Solar Trust Model describes a method by which both the identity and the trustworthiness of the sender of the message can be determined, regardless of whether or not the message was signed by a certificate authority or other entity with which the recipient has a relationship. Because the Solar Trust Model operates in a manner different from other trust models, it is important to precisely define the terms and concepts that are used in the model.

A **Solar Trust Server (STS)** is the primary instrument of implementation of the Solar Trust Model. More precisely, the server is the combination of hardware and software that permits messages to be routed along specified paths, and that permits specific qualities about the message (in particular, trust) to be interpreted. **Trust** is the level of confidence that an STS places in the ability of another entity to meet certain criteria. As with any server, an STS has users. A **User** is anyone or anything that uses an STS to interpret the level of trust of a message, or to send and receive messages with the intent of following the rules of the Solar Trust Model.

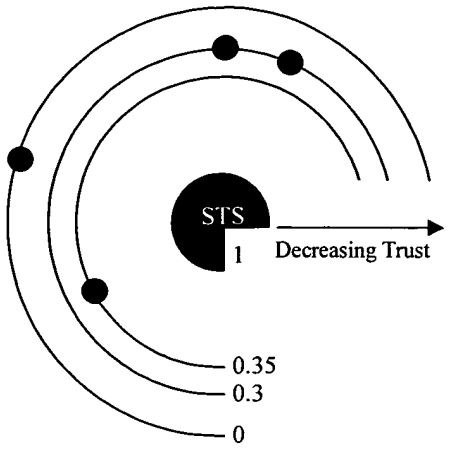


Figure 1. A typical Solar System. Four orbits have been defined in this example. Orbit 0.3 contains two entities. The other orbits contain one entity each. Orbit 1 contains the STS.

An **Entity** is any user, STS, or organization. Entities are grouped into orbits. An **Orbit** is an unordered set of entities. Orbits are labeled with rational numbers. Rational numbers are used for several reasons. First, there are an infinite number of rational numbers and therefore an infinite number of possible labels for orbits. Second, given any two rational numbers, it is always possible to find another rational number between them. Because it is always possible to label an orbit such that its label falls numerically between the labels of two other orbits, and because rational numbers can always be ordered by value, these labels provide a way to create a relative ordering of orbits. It is important to realize, however, that the label of an orbit is really just a name for the orbit, and that orbits do not behave in the same way that numbers do. For example, you cannot add or subtract orbits from one another.

The primary structure upon which the Solar Trust Model is based is called a **Solar System** (see Figure 1). A Solar System consists of an STS, and an ordered set of disjoint orbits. The orbits around an STS are ordered by level of trust. The most trusted (highest valued) orbit in a solar system is orbit 1.0. This orbit contains the STS, since the STS always trusts itself completely. The least trusted orbit in the ordering is designated as orbit 0.0. An entity in orbit 0.0 is considered to be completely untrustworthy.[1]

[1] In contrast with the original approach proposed for the Solar Trust Model in [7], the range of values that can be assigned to orbits in a solar system have been changed from integers that increased from 0 to infinity to rational numbers that decrease from 1.0 to 0.0. This has been done in order to simplify a future implementation.

It is easiest to conceptualize a solar system as analogous to a solar system in space. The solar system has a **Star**, or STS, at its center and is surrounded by concentric orbits that contain other entities. (The analogy to a solar system in space is not exact. For example, in a solar system in space, two objects do not generally share the same orbit, nor are the orbits necessarily concentric.) The set of all Solar Systems is called the **Universe**.

When a person or organization interacts with another entity, it may place some level of confidence or trust in that entity. For example, a person may place a certain level of trust in their friends, a different level of trust in their family, and a third level of trust in their colleagues. An STS is able to represent a level of trust through the use of a **Relationship**. A relationship is a function *f(object, solar system, policy) = orbit*. Given some object, a solar system and a policy, the relationship function maps the object into a specific orbit in that particular solar system using the policy. Relationships are not associative. For example, if STS A has a relationship with STS B, there is no guarantee that STS B has the same relationship, or any relationship all, with STS A.

Relationships come in different forms. For example, while a person may know their friends directly, they may only know the friends of their friends indirectly. For an STS, a **Direct Relationship** is any relationship that an STS has with another entity that the STS has defined using only its own policies and knowledge of that other entity, without relying upon the policies and knowledge of some intermediary third party. Such entities include other servers and users. In keeping with the solar system analogy, any entity with which an STS has established a direct relationship is known as a **Planet**. A **Direct Trust Relationship** is a direct relationship that is based upon the level of trust that an STS has in another entity.

One of the roles of an STS is to determine the level of trust that can be placed in a message, or in the sender of that message. A **Message** is a set of data that is sent from one STS to another. The contents of the message are arbitrary, and a message may contain other messages. When a message is transmitted, it is always sent along a predetermined **Path** (see figure 2). A path is an ordered set of pairs <S, O> such that the following conditions are met:

1) S is an STS.
2) For each S, O is the orbit in the solar system of S that contains the entity from which that S directly received the message.
3) The set contains all of the STSs traversed by the message.
4) The set is ordered by the sequence in which the message traversed those STSs, beginning with the STS that ultimately received the message, and terminating with the STS that originally sent the message.
5) If an STS appears on the path more than once, each instance in which it appears is considered to be distinct.

$S_{Receiving}$	$O_{Receiving}$
⋮	⋮
S_3	O_3
S_2	O_2
$S_{Sending}$	$O_{Sending}$

Figure 2. A Path

Paths are unidirectional. The fact that a path exists from A to B to C does not imply that a path exists from C to B to A. A **Subpath** is any path that can be composed by recursively taking a previously found path and removing the terminating pair in that path. (A path of length *n* has *n-1* subpaths.) An STS operates on a specific kind of path called a **Path of Trust**: This is a path where, with the exception of the original sending STS, all of the STSs in the set have a direct trust relationship with the next STS in the set.

Given that a message has been sent from an entity E to STS R_n through one or more intermediate STSs $R_1 \ldots R_{n-1}$, an **Indirect Relationship** is any relationship that R_n has with E, where the relationship is based explicitly on the combination of policies of each individual STS along the path followed by a message (see figure 3). Any entity with which an STS does not have either a direct or indirect relationship is always considered to be in orbit 0 of that STS.

An **Indirect Trust Relationship** is any indirect relationship that is based explicitly on the combination of policies of each individual STS along a path of trust. STSs dynamically create, change and remove direct and indirect trust relationships with each other. This results in the formation of an ever-changing network of trust relationships.

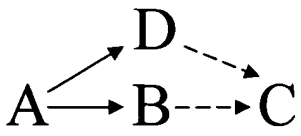


Figure 3. A has a direct relationship with B and D, and two different indirect relationships with C (one through D and one through B).

3. The Solar Trust Model and Ad Hoc Networks

A network of trust relationships between STSs behaves in a very similar manner to an **ad hoc wireless network**. In an ad hoc wireless network, the presence or absence of a unidirectional direct connection between two nodes depends on whether or not the two nodes are within range of each other. By extension, the presence or absence of a unidirectional indirect connection between two nodes depends on whether there exists some set of nodes such that there exists a path between the sending and receiving node where every intermediate node along the path is within range of both an upstream node and a downstream node. Additionally, because the nodes in an ad hoc wireless network can move around, turn off completely, or suddenly turn back on, a transmission path that was valid at one particular moment may not be valid the next moment.

A direct trust relationship from one STS to another can change at any time depending on the criteria that are used by the maintainers of the STS to determine what direct trust relationships the STS should have. Although a direct trust relationship may exist, the relationship may place the entity with which the relationship exists in such a low numbered orbit that the entity is not considered to be sufficiently trusted for a particular purpose. Likewise, the existence of an indirect trust relationship between two nodes depends on whether or not there exists some set of direct relationships between the receiving and sending nodes such that each receiver sufficiently trusts the next sender on the path. Because these relationships can change at any time, and because the policies that interpret these relationships can also change at any time, a path of trust that was valid at one particular moment may not be valid at the next moment.

Because of the similarity in structure between an ad hoc wireless network and a network of trusted paths and STSs, and because the problem of routing data in an ad hoc wireless network is well studied, this paper proposes to adapt the best routing techniques for ad hoc wireless networks to networks of STSs and their trust relationships. Dynamic Source Routing (DSR) was developed to address the problem of routing information on an ad hoc wireless network [8,9]. Performance simulations [11, 12] have demonstrated that the DSR protocol provides superior performance to other protocols for ad hoc networks. Additionally, various optimizations can be performed on the DSR protocol to further improve its performance [10]. This paper uses a modified version of the Dynamic Source Routing protocol, and incorporates both the ideas of using heuristics for load distribution [10], and route caching optimizations [13], that have recently been proposed for DSR. This paper also incorporates a variety of new optimizations that are specific to the Solar Trust Model, but which also might be applied in the future to the optimization of DSR.

4. Solar Trust Model Networking

4.1 Server Initialization

When an STS is first turned on, it has an empty solar system: every orbit is completely empty except for orbit one, which contains the STS itself. The administrator of the STS must establish a policy to assign meaning to one or more orbits and will label these orbits with an appropriate rational number between 0.0 and 1.0.

Direct trust relationships are then assigned to the STS. These may be assigned through some automatic process (for example, the STS might search a database for other STSs that meet some specific criteria), or entered manually. When an STS forms a direct relationship with another entity, the orbit in the STS's solar system that the entity is mapped to depends on the level of trust that the STS places in that entity relative to the levels of trust that have been assigned to the orbits of the STS's solar system. The relationship that an STS has with a particular entity may change over time, and may even be discontinued altogether.

4.2 Trusted Path Discovery

When an STS adds a new direct relationship with another STS, a set of indirect relationships with other STSs is implicitly established through that relationship. However, in order to determine exactly what indirect relationships exist, the STS must perform a path discovery. The idea of a path discovery is for an STS to learn all of the possible trusted paths that meet a certain set of criteria. The meaning of the criteria is up to the operator of the STS, however the acceptable range of values of the criteria is represented using one or more rules. A **Rule** is a function that outputs a binary decision based upon the entity that created it, and some criteria set by that entity. Given the following terms:

Rs = The relationship of the STS to the sender of the message, represented by the path that the message followed.
O = The orbit in which the entity from which the message was directly received is located.
P = The path taken by the message.

Rs(P) = The lowest labeled orbit O' for which a message arriving along path P can be trusted as much as a message originating directly from an entity in O'.
A rule is anything that satisfies these conditions.

Multiple rules can be applied together as a **Rule Set**. A rule set is an ordered list of rules. Each STS maintains a rule set. When an STS wants to transmit a rule set, it places the rule set into a **Rule Set Header** (see figure 4). A rule set header is a set of data that is attached to a message using a specific format. The header contains one or more

Field A: Local Direct Range
Field B: Local Indirect Range
Field C: Maximum Path Length
Field D: Permitted Local Range for Next CA

Figure 4. The Solar Trust Model Rule Set Header.

Rule Sets. The following fields are represented in the Rule Set Header:

- *Field A*: Local Direct Range: Trust all messages that have been directly signed by an STS in an orbit with a label that is greater than or equal to this value.
- *Field B*: Local Indirect Range: Trust messages that have been indirectly signed by an STS in an orbit with a label that is greater than or equal to this value. Do not trust any messages that have been signed by a STS in an orbit with a label below this value.
- *Field C*: Maximum Path Length: A message can be trusted only if the total number of STSs that have signed the message is no greater than this integer value.
- *Field D*: Permitted Local Range for Next STS: A message that has been indirectly signed by an STS inside the local indirect range can be trusted only if it came from an orbit in that STS's system with a label of no less than this value.

For example, let's say that Alice operates an STS, and that Alice does not consider any message that comes from an orbit lower than 0.8 to be sufficiently trustworthy for a particular purpose. Then Alice will set the value of Field A in the rule set for her STS to 0.8. Now let's say that Alice also trusts messages that reached her STS through an indirect relationship. She can specify a value of 0.9 in Field B to guarantee that only messages that arrive through indirect relationships that originate in an orbit with a label of 0.9 or greater will be trusted. If Alice enters a value in Field C, then a message sent to her STS will only be trusted if the length of the path taken by the message is less than or equal to this number. Finally, if Alice's STS has a direct relationship with other STSs, and she knows something about the criteria used by those STSs to determine their own trust orderings, then she may choose to restrict messages that come indirectly from those STSs to certain orbits within the Solar Systems of those STSs by specifying a value for field D.

Rule Sets from many STSs can be concatenated together into a **Composite Rule Set** (see figure 5). A composite rule set is a set of policies that as a group interpret the level of trust that the recipient of a message can place in the path followed by that message. A composite rule set is created by concatenating all of the rule sets for the STSs in the order that the message has passed though them.

Rule set of receiving STS (A)	Rule Set of STS B, with which A has a direct relationship	Rule Set of STS C, with which B has a direct relationship	Rule Set of Sending STS D, with which C has a direct relationship

Figure 5. A Composite Rule Set. The rule set for each STS that a message passes through is added on to the left side of the composite rule set. The complete composite rule set is read from left to right.

4.3 Path Discovery Algorithm

To perform a path discovery, a server sends out a Path Query Message to an STS with which it has a sufficiently trustworthy direct relationship. A path query message is composed of an ordered set of one or more 4-tuples of the form <S, O, N, T>, where S is the identity of the sender of a query, O is the orbit of S that the STS that the query is being forwarded to is in, N is the maximum path length from Field C of the Rule Set Header, and T is an integer representing time in seconds chosen by each S individually. It may also contain additional path queries attached to the end of the message (see figure 6).

S	O	N	T	Piggybacked path queries

Figure 6. Format of a Path Query Message.

A new query is normally initiated either at the time when that relationship is created, or when that relationship becomes sufficiently trustworthy as the result of a change in server policy or a change in the relationship. To minimize the flooding effect that can result from sending out a path query, a server with more than one new or changed direct relationship should space out the transmissions of path query messages to multiple STSs.

Path query messages are not sent to any STS that is not in a sufficiently trustworthy direct relationship, since any message that was received indirectly through that STS would be considered insufficiently trustworthy and would be discarded. When a path query message is received from another STS, it is saved in the STS's **Received Query Cache.**

Two other caches are initially used during path discovery. The first is an **Update Path Cache**. This is a cache of paths, and all of the subpaths of those paths, that an STS has learned of as a result of receiving them in path queries. These paths are used for performing path updates, and also represent possible ways in which the STS could *send* a message to a particular receiver. The set of all paths that the STS has already determined that it trusts is saved in the **Trusted Path Cache**. Put another way, the Trusted Path Cache contains all of the paths along which the STS can *receive* messages in a trusted manner.

Upon receiving a path query message, an STS executes the following algorithm:

1) Decrement the value of N in each 4-tuple in the query message.

2) If the STS has not created or forwarded a path query message recently, add a 4-tuple <S, O, N, T> for itself to the end of the query message.

3) While there exists a 4-tuple in the query message which contains an N with a value of zero:
   a. Define the head of the list as the ordered subset that begins with the 4-tuple at the beginning of the query message and ends with the first 4-tuple that is identified with a value of N = 0.
   b. Define H as the number of members (4-tuples) in the head of the list.
   c. Create a path by taking each pair <S, O> from the 4-tuples in the head of the list in order and adding it to the path.
   d. Save a copy of this path in the STS's Update Path Cache.
   e. Create a list of times $T_1 \ldots T_n$ by taking each T from the 4-tuple in the head of the list in order and adding it to the list of times.
   f. Generate RND = a random rational number between 0 and 1.
   g. While the path has a length > one:
      i. Wait for an amount of time $T_w = T_1 * RND * H$
      ii. Send a copy of the path to the STS identified by the S in the pair at the head of the list. (The transmission should be made out of band, using a standard TCP connection for example.)
      iii. Remove the pair at the head of the list .
      iv. Remove the time T from the head of the list of times.
   h. Remove the head of the list from the path query message.

4) If part of the list in the query message remains, save this list in the current STS's Received Query Cache.

5) If the current STS's Local Direct range contains no planets
   a. Set the value of N in the 4-tuple at the end of the query message = 0.
   b. Repeat step 3.

6) Otherwise, for each STS that is a planet in the current STS's Local Direct range:
   a. Assign the value of the orbit that that planet is in to the O in the 4-tuple at the bottom of the list.
   b. Forward the query to that planet.

This algorithm does several things. First, the STS that initiates the path query will receive a set of paths that describe all sufficiently trusted paths to that STS. The paths that are transmitted to the STS will be of either maximum path length, or of the maximum length that the path could reach before another pair <S,O> could not be added to it. (This would happen because the last STSs to receive the path query did not have a direct relationship through which the indirect relationship could continue (see figure 7)). The STS that initiated the path query can then use these paths to find the set of all trusted paths to it, since any subpath of a sufficiently trusted path that ends at the STS must also be a sufficiently trusted path. All of these paths and subpaths are saved in the trusted path cache. Second, it allows any intermediate STS to piggyback its own path discovery query on top of the initial path discovery query if it needs to do so. Little additional network overhead is added by doing this because an STS that needs to do a path query would do so anyway, and piggybacking allows the STS to use path discovery information that was already propagating through the network (see figure 8). Note that in step 3g, paths of length 1 are not returned to query initiators, because a path of length one indicates a direct relationship, which the initiator must already know about by definition. Third, the time delay built in to the algorithm scales the response time to the query of each node that is sending a path back to the initiator of a query. Because the number of responses to a query can be expected to grow exponentially with each increase in maximum path length, there is a danger that a query initiator, and the physical network that it connects to, will be overloaded with a flood of responses. By setting an appropriate value of T, each query initiator can control the

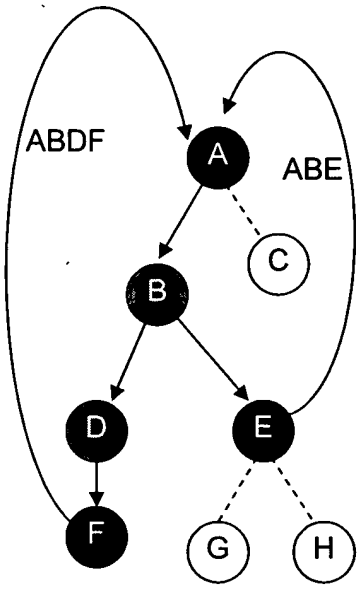


Figure 7. Propagation of a path query: Straight arrows indicate sufficiently trusted direct relationships. Dashed lines indicate insufficiently trusted direct relationships. STS A initiates a query with Maximum Path Length N = 4, and receives paths ABDF and ABE out of band (curved arrows).

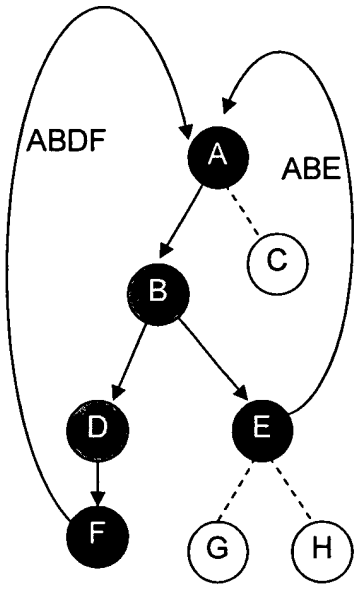


Figure 8. B has a Maximum Path Length of 2. It receives a path from a piggybacked path query from an STS two nodes away. STS A might otherwise have a longer acceptable path, but B's policy truncates that path. Queries added by D or F are not affected by B's maximum path length.

rate at which it receives responses. Furthermore, the use of an independent random number generation by each query responder ensures that the responses that are generated will be distributed evenly over a period of time that increases with the number of likely respondents. Finally, this algorithm guarantees that an STS that initiates a query will never receive a path that is not sufficiently trusted at the time the query passes along the path.

4.4 Path Sorting And Optimization

Once an STS has cached a set of trusted paths, it can determine which of the subset of paths from a particular sender is optimal according to a certain criteria. Since each server defines trust in its own way, each server will maintain a different policy about which orbit to map a particular path into. For a direct path to the STS, no mapping is necessary because the orbit in which a message originated is already known. For an indirect path to the STS, the composite rule set for that path is used by the STS in order to properly perform the mapping. One possible way for the STS to do this would be to start with the orbit from which the message was directly received as a baseline, and then to make adjustments as appropriate. In general, it is expected that the trust in a message that follows a particular path will degrade as the length of the path increases, and as the message passes through orbits of STSs solar systems that are increasingly far from the center of the solar systems.

One effect of performing such a mapping is that all of the paths of trust from a particular sender to the STS will automatically be ordered by relative trust. This makes it easy to compute an **Optimal Path.** An optimal path is a path that produces the most desirable result according to certain criteria when compared to other possible paths. The **Most Trusted Path** therefore is an optimal path, which when evaluated using the composite rule set for that path, will produce a greater level of trust than other paths of trust. Therefore, the most trusted path from a sender to a receiving STS would be the path in the orbit closest to the STS (with a label closest to 1.0). Similar methods can be used to determine a **Least Cost Path,** which is an Optimal Path that minimizes the money, amount of computation, or time that must be expended for a message to travel along a path that will place it into an acceptable orbit of the receiving STS. In some situations, an STS may wish to receive a message from a sender along the path that is optimized for one or more qualities. In other situations,

any valid path from the sender to the receiving STS may be acceptable.

4.5 Path Update Algorithm

As mentioned in the path discovery algorithm, each STS keeps a cache of all paths to it that it knows about. An STS learns of these paths during a path discovery when it receives a path query, and caches them for future reference. If an STS changes a direct relationship with a planet in its solar system, all of the paths that rely on that direct relationship are invalidated. When this occurs, the STS must perform a path discovery on that planet if the planet is still considered to be sufficiently trusted for a particular purpose. (If the planet is no longer considered sufficiently trusted, then there exist no paths through it that could be considered to be sufficiently trusted.) The STS performs this path discovery using the following algorithm:

1) The STS chooses any query from its Received Query Cache and forwards it to the planet with which it has established a new or different relationship.

2) For every other query in the Received Query Cache, the STS:
    a. Removes the pair at the end of the query list. (This is the pair that corresponds to the STS's own query. That query was sent out in the previous step.)
    b. Forwards the query to the planet with which it has established a new or different relationship.

This algorithm updates the indirect relationships of any STS that has discovered paths that run through this STS, as well as the relationships of the STS that has changed its policy or relationships. As an optimization, it may be helpful to set a "do not propagate" bit to notify STSs that receive the query that they should not add their own queries on to the update query unless it is absolutely necessary (i.e., they also need to perform an update and are ready to do so). This would prevent update queries from triggering a new query flood when one is not needed.

4.6 Sending a message

When one STS (the sender) needs to send a message to another STS (the receiver), it must know which path that message must follow in order to reach the receiver. The receiver does not care what path the message takes from the sender. Regardless of the path that the message takes, the receiver will still have to make a judgment about the level of trust that should be assigned to the message using the Rule Set Header. However, it is in the best interests of the sender for the path taken by the message to be as trusted as possible. The sender can determine this path in one of two ways:

1) If the sender has a path to the receiver in its Update Path Cache, and this path was received recently, then it is likely that this path is still valid. There is no guarantee that this path is any more or less trusted than any other path, but sending a message along this path will probably result in the receiver considering the message to be sufficiently trusted. The longer that this path remains in the cache, the less likely it is that this path will still be trusted.

2) If the sender does not wish to use a path in its Update Path Cache for any reason, the sender can send an out of band query (most likely using a TCP based message) to the receiver identifying itself and asking what path the message should take. The receiver can then reply with a suitable path, and the sender will send out the message along that path.

When a sender sends a message, that message is combined with an appropriate path to form a **Message Packet**. A message packet is a data structure consisting of a path followed by a signed block of data consisting of a rule set header, an orbit, and a message (see figure 9). The following algorithm is used when sending a message:

1) If a path is not attached to the front of the message (i.e. if the message is not encapsulated in a message packet), determine the appropriate path for the message to take.

2) If the current STS is not the receiver:
    a. Remove the STS from the path.
    b. Create a header composed of:
        i. The STS's rule set header.
        ii. The orbit from which the message was received. (In the case where the STS is the original sender of the message, if the message comes from that STS itself, the orbit must equal one since an STS must trust itself completely. If the STS is acting on behalf of a user with whom it has a relationship, the orbit is the same as the value of the relationship.)
        iii. The identity of the STS from which the message was received.
    c. Sign the combined message and rule set header. The signed message and rule set header are now considered to be the message in the message packet.
    d. Forward the message packet to the next STS on the path.

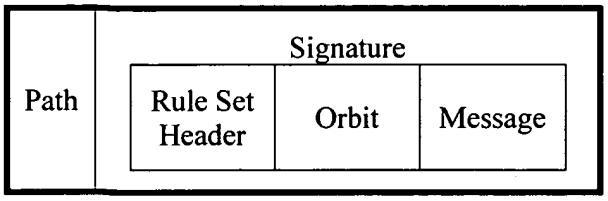


Figure 9. Format of a Message Packet.

4.7 Receiving and parsing a message

Once the message has reached the receiver, the receiver uses the information contained in the message to determine the level of trust that can be assigned to the message. The receiver builds a composite rule set header and a path by recursively reading the rule set, orbit, and identification information of each message, and verifying the message signature. The following algorithm is used to do this:

1) Begin a composite rule set by adding the rule set of the receiving server to an empty composite rule set.

2) Begin a path of trust by adding a pair <S, O> where S is the identity of the receiving server, and O is the orbit in the current server's solar system that contains the planet from which the message was directly received.

3) Do the following until the current message does not contain any smaller messages:
   a. Verify the signature on the message. If the signature is not valid, the message cannot be trusted. Stop processing on the message.
   b. Remove the rule set header from the message, and add it to the end of the current composite rule set for this message.
   c. Create a pair <S, O> where S is the server identity on the message and O is the orbit label on the message.

4) Add the pair to the end of the path of trust.

Once this process is completed, the server can make a judgment about whether, and by how much, the message should be trusted, by applying the rules in the composite rule set, in order, to the path that the message followed. Any message that is not sufficiently trusted will automatically be eliminated by the application of the composite rule set. The composite rule set is always parsed starting with the rules added by the receiving STS, and ending with the rules added by the sending STS. If a message is sufficiently trusted, the STS can make a judgment about the degree to which that message can be trusted by using its own policies to interpret the information contained in the path and the composite rule set.

4.8 Path cache updates

When an STS receives a message packet, it can use what it learns about the path contained in that message packet to update its path cache. If a message is deemed to be sufficiently trusted after following a path that is not in the path cache, that path is added to the path cache. Additionally, all subpaths from the receiver along the path to the sender can also be added to the path cache. If, on the other hand, the STS determines that the message followed a path that is not sufficiently trusted, and that the path that it followed was in the cache, then all paths in the cache that contain the untrusted path as a subpath must be deleted.

5. Direct Relationship Example

Imagine that Alice and Bob are surgeons in a hospital in Arlington. Alice's hospital runs an STS named Alpha. Based on the policy of the hospital, Alpha has placed Alice in orbit 0.85 and Bob in orbit 0.75 for trust issues relating to medical decisions.

Charlie is a surgical resident at the hospital. Dave, a patient with severe symptoms comes in to see Charlie. It is late at night, and both Alice and Bob are at home. The patient degrades quickly, leaving Charlie no choice but to perform an emergency surgery on the patient. A complication arises that exceeds Charlie's level of

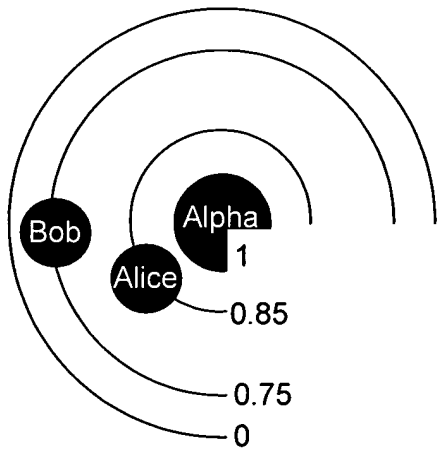


Figure 10. Alice's advice is considered to be more trustworthy than Bob's, because Alice is in an orbit with a higher numbered label.

expertise. Although Alice and Bob are both too far away to make it to the hospital in time to save the patient, Charlie can communicate with Alice and Bob through e-mail, using Alpha for authentication. He immediately sends a message to both of them asking how to deal with the complication. Alice replies that he must inject a drug to lower the patient's blood pressure. Bob replies that he must inject a drug to raise the patient's blood pressure. Following the wrong advice will certainly kill the patient, but which advice should he follow? Alpha assigns a greater level of trust to Alice's message then to Bob's message because Alice is in an orbit with a higher label than Bob. Therefore, Charlie chooses to follow Alice's advice. (See figure 10.)

6. Indirect Relationship Example

Several months later, Dave has moved to Denver when he has a relapse and must immediately have another surgery. Doris, Dave's surgeon in Denver, does not know Dave's medical history, and must obtain Dave's patient records from Arlington. She sends an out of band request for the records to Alpha through Delta, the STS of the Denver hospital. The Arlington hospital has a policy that describes the level of trust required for patient records to be sent to a third party. The Arlington hospital has no direct relationship with the Denver hospital, and therefore Delta is not in the solar system of Alpha. If the Arlington hospital's policy only permitted patient records to be sent to other hospitals or doctors with which it has a sufficiently strong direct relationship, then Alpha would not permit the patient records to be sent to Delta because Alpha does not have a direct relationship with Delta.

Fortunately, the Arlington hospital's policy also permits patient records to be transferred to other hospitals or doctors if a sufficiently trusted indirect relationship exists with that hospital or doctor. Alpha has a direct relationship labeled 0.6 with Bravo, the STS at the Boston hospital. Because the Boston hospital has worked closely with the Denver hospital in the past, Bravo has placed Delta in an orbit labeled 0.9. Based on the Denver hospital's own policy, Doris is in an orbit with a label of 0.638 in Delta's solar system. Alpha checks its trusted path cache, and determines that a path of trust to Delta has previously been computed using the path discovery algorithm. Furthermore, it also determines that a message sent along this path is likely to be sufficiently trusted to meet the requirements for the release of confidential patient records. Thus, Alpha replies out of band with the path from Delta to itself.

Delta creates a message packet with Doris' request, its relationship with Doris, and the appropriate rule set header, path header and signature. It forwards this message packet to Bravo, which repeats the process and forwards the message on to Alpha. Alpha determines that is the receiving STS. It creates a composite rule set and checks the signatures of Bravo and Delta. If the composite rule set says that Bravo and Delta are sufficiently trusted, and that Doris is in a sufficiently trusted orbit of Delta, then Alpha will permit Dave's patient records to be forwarded to the Denver hospital. Otherwise, Alpha will not forward the patient records and may choose to reply with a refusal.

7. Conclusion

The Solar Trust Model provides a method by which an entity can authenticate an entity with which it does not have a pre-existing relationship. It can also determine the extent to which it can trust that other entity, without the limitations inherent in traditional Public Key Infrastructures and trust models. A variety of enhancements to the model have been presented. The objects in the model have been better defined from [7]. A suite of algorithms has been presented for path discovery, path sorting and optimization, path updating, message receiving and path parsing. Some of these of algorithms are loosely based on the Dynamic Source Routing algorithm. These enhancements provide the missing elements that are necessary for the implementation of the Solar Trust Model. Implementation of the model could ultimately lead to a distributed, user-centric, and universal method of determining indirect trust relationships.

8. Future work

The use of additional optimizing strategies such as server capability based query and path routing are currently being investigated, as are additional path distribution mechanisms such as partial path advertisements. Future research directions include a simulation of these algorithms to determine their scalability, and to identify further optimizations that can be made to them. The implementation and testing of these algorithms in actual STSs is planned. Additionally, methods of preventing spoofing of query sender identities will be identified, in order to ensure that STSs can not be used for denial of service attacks.

References

[1] International Telecommunication Union. "Information Technology – Open Systems Interconnection – The Directory: Authentication Framework," *ITU-T Recommendation X.509*, pages 7-17. 1995.

[2] Kent, S. "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management." Internet Report, RFC 1422, February 1993.

[3] L. Lpez, J. Carracedo. "Hierarchical Organization of Certification Authorities for Secure Environments." Proceedings of the Symposium on Network and Distributed System Security, Februrary, 1997, San Diego, CA.

[4] Biba, K. "Integrity Considerations for Secure Computer Systems." U.S. Air Force Electronic Systems Division Technical Report 760372, 1977.

[5] Intel Corporation. "Common Security Services Manager Trust Policy Interface (TPI) Specification Draft for Release 1.2", page 7, March 1997.

[6] Young, A. Cicovic, N.K. and Chadwick, D. "Trust models in ICE-TEL". Proceedings. 1997 Symposium on Network and Distributed System Security. (pp. 122-133), Feb. 1997.

[7] M. Clifford, C. Lavine, and M. Bishop, "The Solar Trust Model: Authentication Without Limitation," Proceedings of the 14th Annual Computer Security Applications Conference, 1998, pp. 300-307.

[8] Broch, J., Johnson, D., and Maltz, D. "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks". In Internet draft, IETF Mobile Ad Hoc Networking Working Group (Dec. 1998).

[9] David B. Johnson and David A. Maltz. "Dynamic Source Routing in Ad Hoc Wireless Networks". In "Mobile Computing", edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publishers, 1996.

[10] L. Wang, L. Zhang, O. Yang, Y. Shu and M. Dong, "Multipath Source Routing in Wireless Ad Hoc Networks", 2000 Canadian Conference on Electrical and Computer Engineering, Volume 1, p.479-483. 2000

[11] Meggers, J. and Filios, G., "Multicast communication in 'ad hoc' networks". 48th IEEE Vehicular Technology Conference, Vol.1 (pp. 372 –376) 1998.

[12] Das, S.R., Perkins, C.E. and Royer, E.M "Performance comparison of two on-demand routing protocols for ad hoc networks". IEEE INFOCOM 2000. Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Vol.1 Page(s): 3 –12, 2000.

[13] Marina, M.K.and Das, S.R. "Performance of route caching strategies in Dynamic Source Routing". 2001 International Conference on Distributed Computing Systems Workshop, (pp. 425 – 432), 2001

[14] Bodeau, D. and Connolly, J. Findings from the May 2001 Workshop on Information-Security-System Rating and Ranking, as presented at the 17th Annual Computer Security Applications Conference in the tutorial on "Information Assurance 'Metrics'", Dec 11, 2001. http://www.acsac.org/2001/tutorials.html#tut7